

LAPORAN PROJECT UAS

DATABASE



UBAYA
UNIVERSITAS SURABAYA

Disusun Oleh:

KELOMPOK 23

160821002 Benny Ebenhaezer

160721046 Ryan Hadi

160421098 Antony Kariono

160421105 Komang Wahyu Sri Adijaya

160421109 Putu Pramodia Suka Pratama

FAKULTAS TEKNIK UNIVERSITAS SURABAYA

PROGRAM STUDI TEKNIK INFORMATIKA

TAHUN AJARAN 2022/2023

Form Pembagian Tugas

Pembuatan ER Diagram

- Komang Wahyu Sri Adijaya (160421105)
- Putu Pramodia Suka Pratama (160421109)

Pembuatan User Interface

- Benny Ebenhaezer (160821002)
- Ryan Hadi (160721046)
- Antony Kariono (160421098)
- Komang Wahyu Sri Adijaya (160421105)
- Putu Pramodia Suka Pratama (160421109)

Pembuatan & Penerapan Class

- Putu Pramodia Suka Pratama (160421109)
- Komang Wahyu Sri Adijaya (160421105)

Pembuatan Laporan

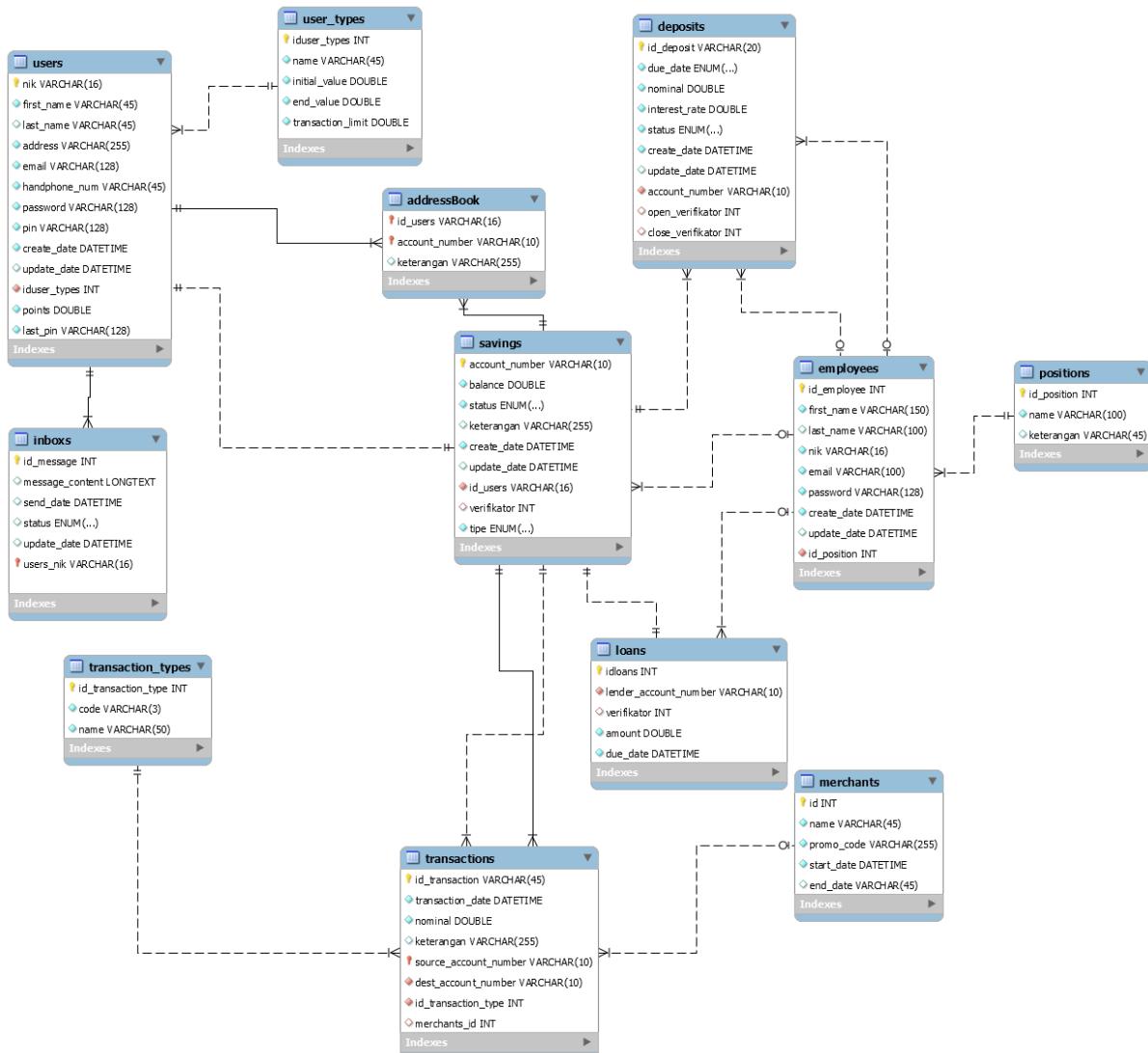
- Komang Wahyu Sri Adijaya (160421105)
- Putu Pramodia Suka Pratama (160421109)

Pembuatan Slide Powerpoint

- Benny Ebenhaezer (160821002)
- Ryan Hadi (160721046)
- Antony Kariono (160421098)

1 Milestone Tahap 1

1.1 ERD dan Fitur Tambahan



Fitur tambahan :

A. Catatan pergantian pin.

Kami menambah atribut baru di dalam entitas users bernama last_pin. Atribut ini berfungsi untuk mencatat pin yang telah digantikan. Maksimal pin lama yang disimpan hanya 1 , jadi pengguna tidak akan bisa memakai pin yang sama dengan 1 pin sebelumnya ketika mengganti pin. Ketika pengguna membuat akun, data kosong akan dimasukkan ke dalam atribut last_pin. Ketika pengguna mengganti pin, pin baru akan menggantikan angka dari atribut pin, sedangkan nilai dari last_pin akan diisi dengan pin yang tergantikan.

B. Limit transaksi berdasarkan user

Kami menambah atribut baru yaitu `transaction_limit` di dalam entitas `user_type`. Masing - masing tipe pengguna akan memiliki *limit* transaksi yang berbeda. Jika user mentransfer melebihi *limit* transaksi dari tipe *user* mereka, maka aplikasi akan memberi info bahwa transaksi telah melebihi batas dan pengguna harus menurunkan nominal transaksi mereka.

C. Pinjaman bank.

Kami menambah entitas baru bernama `loans`. Entitas ini akan mencatat daftar pinjaman yang telah dibuat oleh pengguna. Relasi entitas ini adalah one to one dengan entitas `savings` dan one to many dengan entitas `employee`. 1 akun tabungan hanya mampu meminjam 1 kali saja di dalam satu waktu , dan 1 `employee` mampu memverifikasi banyak pinjaman. Untuk pembayaran pinjaman . akan melalui sebuah akun milik DiBa sendiri yang tersimpan di dalam database. Entitas `loans` juga memiliki atribut bernama `due_date`, yang berfungsi untuk mencatat waktu tenggat dari hutang yang dipinjam. Waktu tenggat untuk semua peminjaman hutang adalah 6 bulan dari tanggal peminjaman. Jika pengguna tidak membayar hutang mereka tepat waktu maka konsekuensinya adalah pengguna tidak akan bisa membuka deposito dan melakukan transaksi ke pengguna DiBa yang lain. Pengguna bisa membuka deposito dan melakukan transaksi setelah hutang yang telat dibayar dilunasi.

1.2 Settings

A. Pengaturan Informasi Database

	Name	Type	Scope	Value
	DbServer	string	User	localhost
	DbName	string	User	newdiba
	DbUsername	string	User	root
*	DbPassword	string	User	

Pengaturan informasi database dibuat dalam sebuah file *settings* yang bernama “db.settings”. Pengaturan ini berisi informasi tentang server, nama, username, dan password dari database yang digunakan.

B. Pengaturan Informasi Tabungan

	Name	Type	Scope	Value
	SavingInterest	double	>User	0.036
	Tax	double	>User	0.1
	SatuBulan	double	User	0.03
	TigaBulan	double	User	0.05
	EnamBulan	double	User	0.06
	SatuTahun	double	User	0.08
	DuaTahun	double	User	0.09
	TigaTahun	double	User	0.1
	Penalty	double	User	0.05

Pengaturan informasi tabungan dibuat dalam sebuah file *settings* yang bernama “savingInformation.settings”. Pada pengaturan ini pengguna dapat merubah:

- SavingInterest = suku bunga tabungan reguler
- Tax = pajak
- SatuBulan = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 1 bulan
- TigaBulan = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 3 bulan
- EnamBulan = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 6 bulan
- SatuTahun = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 1 tahun
- DuaTahun = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 2 tahun
- TigaTahun = suku bunga tahunan tabungan deposito yang memiliki waktu jatuh tempo 3 tahun
- Penalty = jumlah persen pengurangan dari nilai deposito jika mencairkan deposito sebelum waktu jatuh tempo

1.3 CRUD

1.3.1 CRUD Pengguna

A. Class User

```
public class User
{
    private string nik;
    private string firstName;
    private string lastName;
    private string address;
    private string email;
    private string handphoneNumber;
    private string password;
    private string pin;
    private DateTime createDate;
    private DateTime updateDate;
    private double points;
    private UserType userType;
    private string lastPin;
```

Untuk membuat CRUD pengguna kami membuat sebuah class bernama Users. Di dalam class Users ini terdapat 13 *data member*.

B. Create User

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

```
1 reference
public static void AddData(User user)
{
    string command = $"INSERT INTO users(nik, first_name, last_name, address, email, " +
        $" handphone_num, password, pin, create_date, update_date, iduser_types, points, " +
        $" last_pin) VALUES({user.Nik}', '{user.FirstName.Replace("'", "\\'')}', '{user.LastName.Replace("'", "\\'')}','" +
        $" '{user.Address.Replace("'", "\\'')}', '{user.Email}', '{user.HandphoneNumber}', SHA2C'{user.Password}', 512)'," +
        $" SHA2C'{user.Pin}', 512), '{user.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", " ")}, {user.UserType.IdType}, " +
        $"'{user.UpdateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE").Replace("Z", " "))}', {user.UserType.IdType}, " +
        $"'{user.Points}, SHA2C'{user.LastPin}', 512);";
    Connection.ExecuteDML(command);
}
```

Untuk bagian *Create*, agar program mampu membuat objek baru dari *class* Users kami membuat satu *method* baru bernama AddData. AddData akan memanggil *method* bernama ExecuteDML dari *class* Connection untuk menjalankan perintah DML agar data user bisa dimasukkan ke dalam database. *Method* AddData akan dijalankan di form Register. *Form* ini akan dipanggil ketika pengguna ingin membuat akun baru DiBa.

C. Read User

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- FetchDataById
- ReadData

- LoginCheck
- PinCheck

FetchDataByID

```
13 references
public static User FetchDataByID(string nik)
{
    string sql = $"SELECT * FROM users as u INNER JOIN user_types as ut ON u.iduser_types = ut.iduser_types WHERE u.nik = '{nik}';";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if (hasil.Read() == true)
    {
        UserType userType = new UserType(int.Parse(hasil.GetValue(13).ToString()),
            hasil.GetValue(14).ToString(), double.Parse(hasil.GetValue(15).ToString()),
            double.Parse(hasil.GetValue(16).ToString()), double.Parse(hasil.GetValue(17).ToString()));
        DateTime? updateDate;
        if (hasil.GetValue(9).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(9);
        }
        User user = new User(hasil.GetValue(0).ToString(), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString(),
            hasil.GetValue(3).ToString(), hasil.GetValue(4).ToString(), hasil.GetValue(5).ToString(),
            hasil.GetValue(6).ToString(), hasil.GetValue(7).ToString(), (DateTime)hasil.GetValue(8),
            updateDate, double.Parse(hasil.GetValue(11).ToString()), userType, hasil.GetValue(12).ToString());
        return user;
    }
    else
    {
        return null;
    }
}
```

Method FetchDataByID akan berfungsi untuk mengambil data dari *user* yang login. Data yang diambil berasal dari database. *Method* ini akan digunakan di beberapa *class* seperti Saving, AddressBook dan Inbox.

ReadData

```
1 reference
public static List<User> ReadData()
{
    string sql = "SELECT * FROM users as u INNER JOIN user_types as ut ON u.iduser_types = ut.iduser_types;";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    List<User> listUsers = new List<User>();
    while (hasil.Read() == true)
    {
        UserType userType = new UserType(int.Parse(hasil.GetValue(13).ToString()), hasil.GetValue(14).ToString(),
            double.Parse(hasil.GetValue(15).ToString()), double.Parse(hasil.GetValue(16).ToString()), double.Parse(hasil.GetValue(17).ToString()));
        DateTime? updateDate;
        if (hasil.GetValue(9).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(9);
        }
        User user = new User(hasil.GetValue(0).ToString(), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString(),
            hasil.GetValue(3).ToString(), hasil.GetValue(4).ToString(), hasil.GetValue(5).ToString(), hasil.GetValue(6).ToString(),
            hasil.GetValue(7).ToString(), (DateTime)hasil.GetValue(8), updateDate, double.Parse(hasil.GetValue(11).ToString()),
            userType, hasil.GetValue(12).ToString());
        listUsers.Add(user);
    }
    return listUsers;
}
```

Method ini akan mengeluarkan *list* objek pengguna, yang diambil dari database.

LoginCheck

```
2 references
public static User LoginCheck(string emailOrNumber, string password)
{
    string sql = $"SELECT * FROM users as u INNER JOIN user_types as ut ON u.iduser_types = ut.iduser_types WHERE (u.email= '' + 
    $"{emailOrNumber}' OR u.handphone_num = '{emailOrNumber}') AND u.password = SHA2('{password}', 512);";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    while (hasil.Read() == true)
    {
        UserType userType = new UserType(int.Parse(hasil.GetValue(13).ToString()), hasil.GetValue(14).ToString(),
        double.Parse(hasil.GetValue(15).ToString()), double.Parse(hasil.GetValue(16).ToString()),
        double.Parse(hasil.GetValue(17).ToString()));

        DateTime? updateDate;
        if (hasil.GetValue(9).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(9);
        }

        User user = new User(hasil.GetValue(0).ToString(), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString(),
        hasil.GetValue(3).ToString(), hasil.GetValue(4).ToString(), hasil.GetValue(5).ToString(), hasil.GetValue(6).ToString(),
        hasil.GetValue(7).ToString(), (DateTime)hasil.GetValue(8), updateDate, double.Parse(hasil.GetValue(11).ToString()),
        userType, hasil.GetValue(12).ToString());
    }
    return user;
}
return null;
}
```

Method LoginCheck berfungsi untuk melakukan pengecekan terhadap pengguna yang *login*, apakah data yang dimasukkan sudah benar atau belum.

PinCheck

```
public static bool PinCehck(Saving saving , string pin)
{
    string sql = "select * from users " +
                "where nik = '" + saving.User.Nik + "' " +
                "and pin = SHA2('"+ pin + "','512) ";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    bool check;

    if(hasil.Read()== true)
    {
        check = true;
    }
    else
    {
        check = false;
    }

    return check;
}
```

Method ini berguna untuk melakukan pengecekan apakah pin yang dimasukkan ketika melakukan transaksi benar atau salah.

D. Update User

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData
- UpdateCredential
- GantiPin

UpdateData

```
1 reference
public static void UpdateData(User user)
{
    string sql = $"UPDATE users SET first_name = '{user.FirstName.Replace("", "\'")}', last_name = '{user.LastName.Replace("", "\'")}', " +
        $" address = '{user.Address.Replace("", "\'")}', email = '{user.Email}', handphone_num = '{user.HandphoneNumber}', create_date = " +
        $" '{user.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', update_date = " +
        $" '{user.UpdateDate?.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', iduser_types = {user.UserType.IdType}, " +
        $" points = {user.Points} WHERE nik = '{user.Nik}'";
    Connection.ExecuteDML(sql);
}

1 reference
public static void UpdateData(User user, string newPass) // untuk update Password
{
    string sql = $"UPDATE users SET first_name = '{user.FirstName.Replace("", "\'")}', last_name = '{user.LastName.Replace("", "\'")}', " +
        $" address = '{user.Address.Replace("", "\'")}', email = '{user.Email}', password = SHA2('{newPass}', 512), handphone_num = " +
        $" '{user.HandphoneNumber}', create_date = '{user.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $" update_date = '{user.UpdateDate?.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', iduser_types = {user.UserType.IdType}, " +
        $" points = {user.Points} WHERE nik = '{user.Nik}'";
    Connection.ExecuteDML(sql);
}
```

Untuk bagian *update* , kami membuat *method* bernama UpdateData di *class* Users. *Method* ini meminta parameter objek dari *class* User. *Method* ini akan membuat perintah sql *update* bagi user yang memiliki nomor NIK yang sama dengan objek user yang dijadikan parameter. Lalu perintah sql tersebut akan dijalankan oleh *method* dari *class* Connection , yaitu ExecuteDML.

UpdateCredential

```
1 reference
public static void UpdateCredential(string credentialName, string credentialValue, User user)
{
    string sql = $"UPDATE users SET {credentialName} = SHA2('{credentialValue}', 512) WHERE nik = '{user.Nik}'";
    Connection.ExecuteDML(sql);
}
```

Method ini digunakan untuk mengubah data penting pengguna di dalam database.

GantiPin

```
1 reference
public static void GantiPin(User user, string pinBaru)
{
    string sql = "select * from users " +
        "where pin = sha2('" + pinBaru + "", 512) and nik = '" + user.Nik + "'";
    MySqlDataReader result = Connection.ExecuteNonQuery(sql);
    if(result.Read() == true)
    {
        throw (new ArgumentException("Pin yang anda masukkan sama dengan pin saat ini"));
    }
    else
    {
        string sql2 = "select * from users " +
            "where last_pin = sha2('" + pinBaru + "", 512) and nik = '" + user.Nik + "'";
        MySqlDataReader result2 = Connection.ExecuteNonQuery(sql2);
        if(result2.Read() == true)
        {
            throw (new ArgumentException("Pin yang anda masukkan sama dengan pin sebelumnya"));
        }
        else
        {
            string sql3 = "update users " +
                " set last_pin = pin, " +
                " pin = sha2('" + pinBaru + "", 512) " +
                " where nik = '" + user.Nik + "'";
            Connection.ExecuteDML(sql3);
            throw (new ArgumentException("Pergantian Pin berhasil"));
        }
    }
}
```

Method GantiPin akan digunakan ketika pengguna ingin mengganti pin. *Method* ini akan melakukan 2 kali pengecekan sebelum pin diganti. Pengecekan pertama adalah apakah pin baru sama

dengan pin yang lama. Pengecekan kedua adalah apakah pin baru sama dengan pin sebelumnya. Jika pin baru tidak sama dengan pin sekarang dan pin sebelumnya maka program akan mengganti angka pin sekarang dengan pin baru dan angka dari atribut last_pin dengan pin yang tergantikan.

E. Delete User

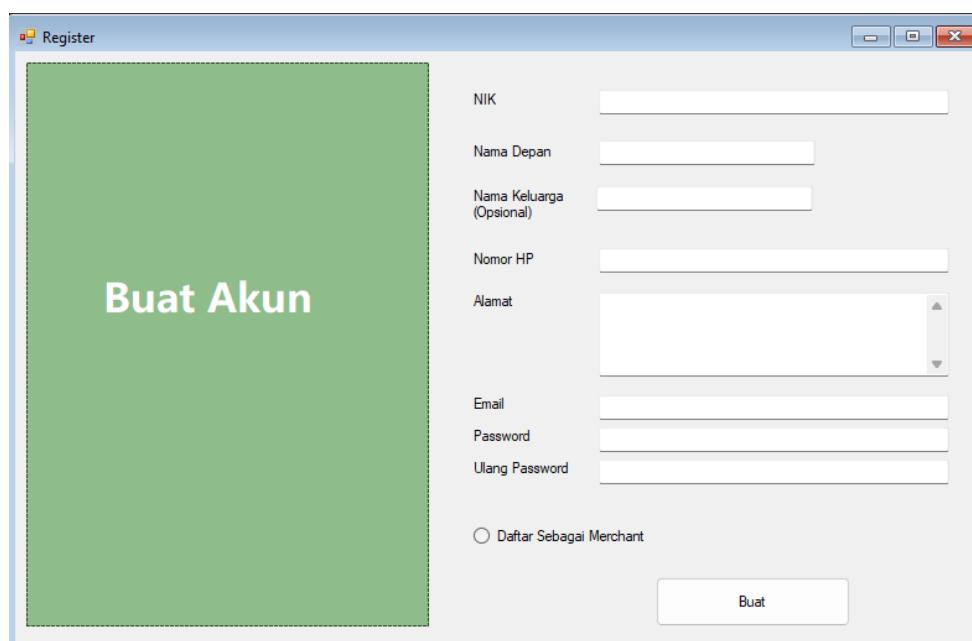
Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- DeleteData

```
0 references
public static bool DeleteData(User user)
{
    string command = $"DELETE FROM users WHERE nik = {user.Nik}";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

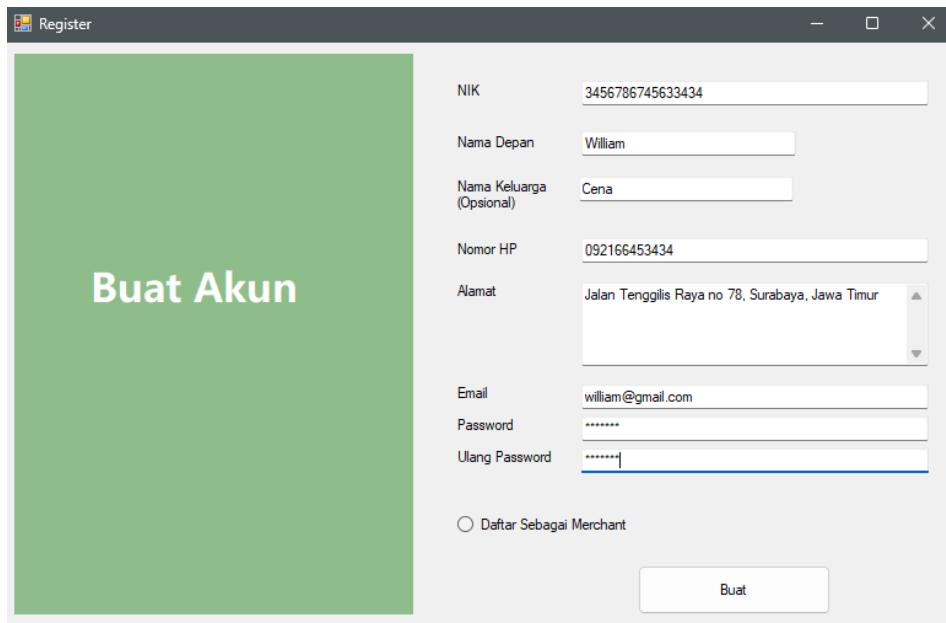
Untuk bagian *delete* , kami membuat *method* bernama DeleteData di *class User*. *Method* ini akan menghapus data pengguna di dalam database berdasar objek pengguna yang dimasukkan ke dalam *argument method*.

F. FormRegister

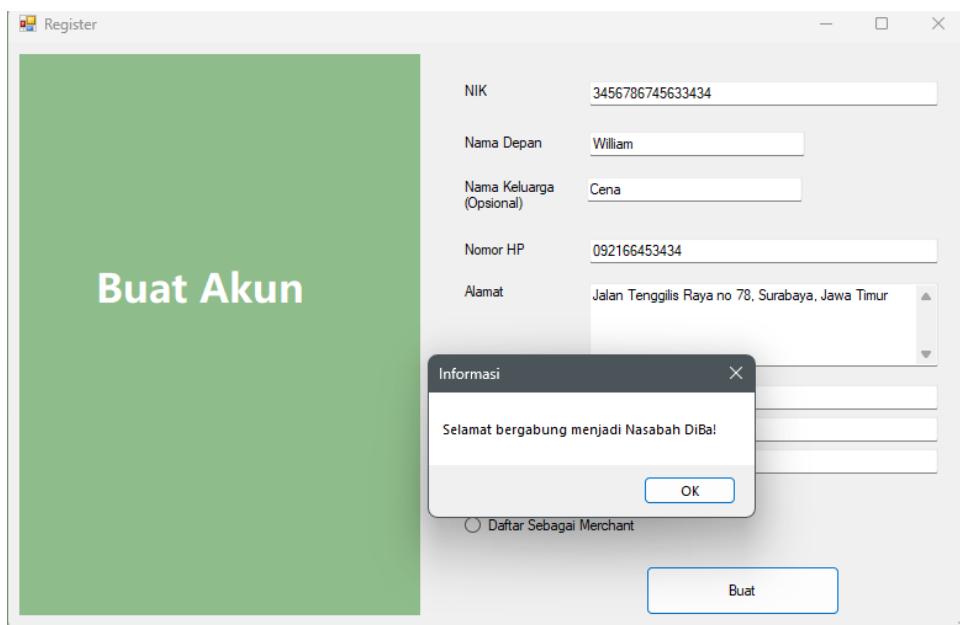


Form ini digunakan ketika pengguna membuat akun baru DiBa. Ketika pengguna membuat akun baru, *Form* ini otomatis juga akan membuat akun tabungan yang baru untuk pengguna. Jika

pengguna mencentang *radio button* bernama “Daftar Sebagai Merchant” maka tabungan yang dibuat akan bertipe *merchant*. Jika tidak maka tipe tabungan adalah *basic*.



Gambar diatas merupakan proses pengguna membuat akun baru DiBa.



Program akan memberitahu pengguna jika pendaftaran berhasil.

```

string tipe = "basic";
List<UserType> listOfTypeUserTypes = new List<UserType>();

1 reference
private void buttonBuatAkun_Click(object sender, EventArgs e)
{
    if (textBoxPassword.Text == textBoxUlangPassword.Text && textBoxNik.TextLength == 16)
    {
        try
        {
            UserType userType = UserType.FetchDataByID(1);
            User newUser = new User(textBoxNik.Text, textBoxNamaDepan.Text, textBoxNamaKeluarga.Text, textBoxAlamat.Text, textBoxEmail.Text, textBoxNoHp.Text,
            textBoxPassword.Text, "", DateTime.Now, DateTime.Now, 0, userType, "");
            User.AddData(newUser);
            MessageBox.Show("Selamat bergabung menjadi Nasabah DiBa!", "Informasi");

            Saving newSaving = new Saving(Saving.GenerateID(newUser), 0, "Unverified", "DiBa saving account", DateTime.Now, null, newUser, null, tipe);
            Saving.AddData(newSaving);
            MessageBox.Show("Selamat bergabung menjadi Nasabah DiBa!", "Informasi");

            this.Close();
        }
        catch (Exception exception)
        {
            MessageBox.Show($"Gagal registrasi! Pesan kesalahan: {exception.Message}", "Error");
        }
    }
    else
    {
        MessageBox.Show("Password harus sama beserta NIK harus 16 digit !");
    }
}

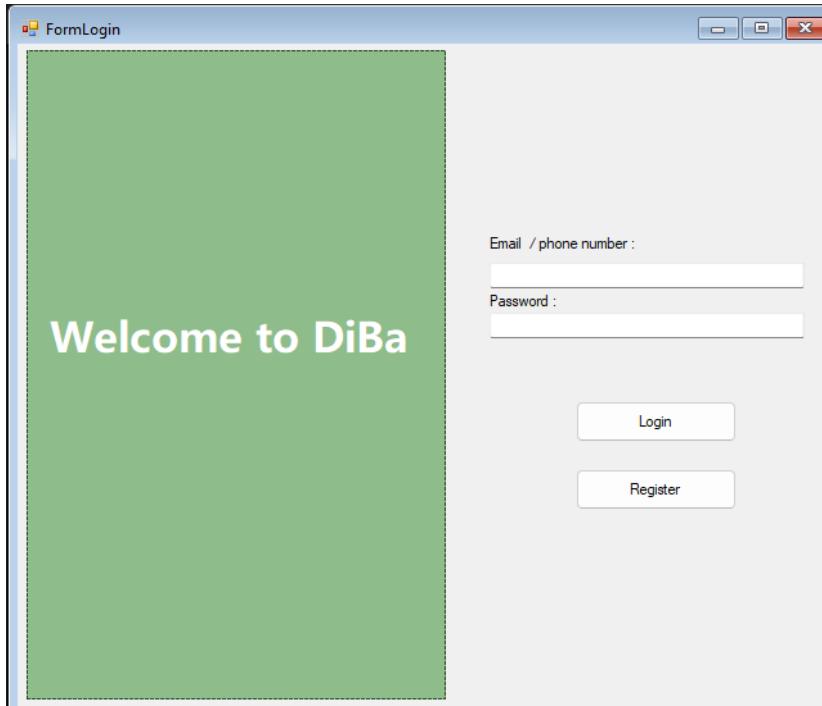
1 reference
private void FormRegister_Load(object sender, EventArgs e)
{
    listOfTypeUserTypes = UserType.ReadData();
}

1 reference
private void radioButtonMerc_CheckedChanged(object sender, EventArgs e)
{
    tipe = "merchant";
}

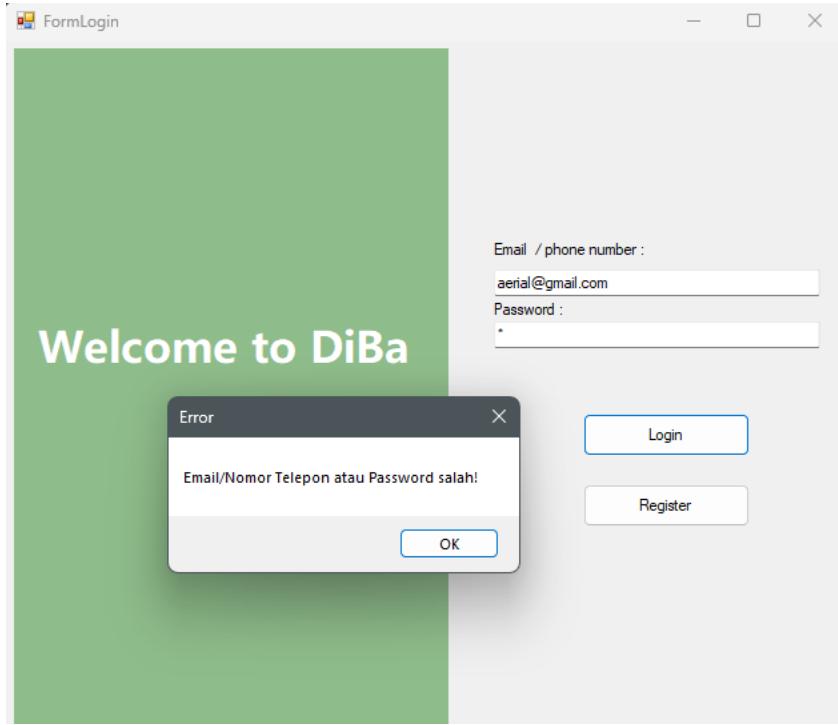
```

Berikut merupakan *code* dari *form* ini.

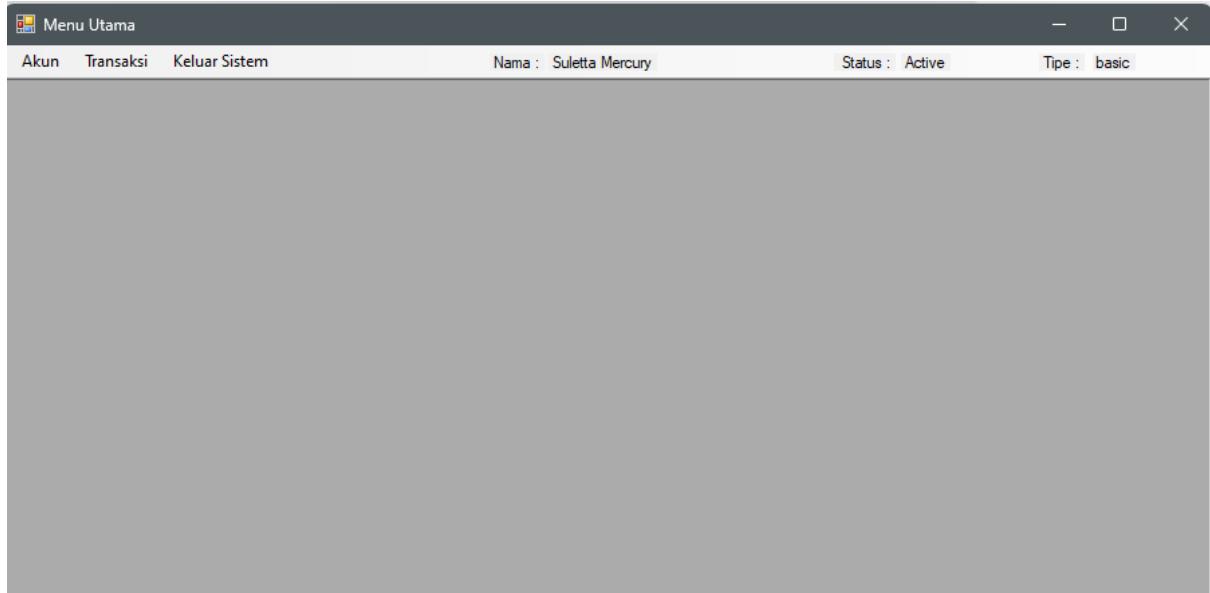
G. FormLogin



Pengguna melakukan *login* melalui FormLogin. Pengguna melakukan *login* dengan memasukkan email atau nomor HP beserta password.



Program akan memberi tahu pengguna jika password atau email/nomor HP yang dimasukkan salah.



Jika *login* berhasil maka pengguna akan masuk ke *form* utama.

```

private void buttonLogin_Click(object sender, EventArgs e)
{
    try
    {
        Connection connection = new Connection();
        User user = User.LoginCheck(textBoxEmailPhone.Text, textBoxPassword.Text);
        if(user is null)
        {
            Employee employee = Employee.LoginCheck(textBoxEmailPhone.Text, textBoxPassword.Text);
            if (employee is null)
            {
                MessageBox.Show("Email/Nomor Telepon atau Password salah!", "Error");
            }
            else
            {
                FormMain formMain = (FormMain)this.Owner;
                formMain.employee = employee;

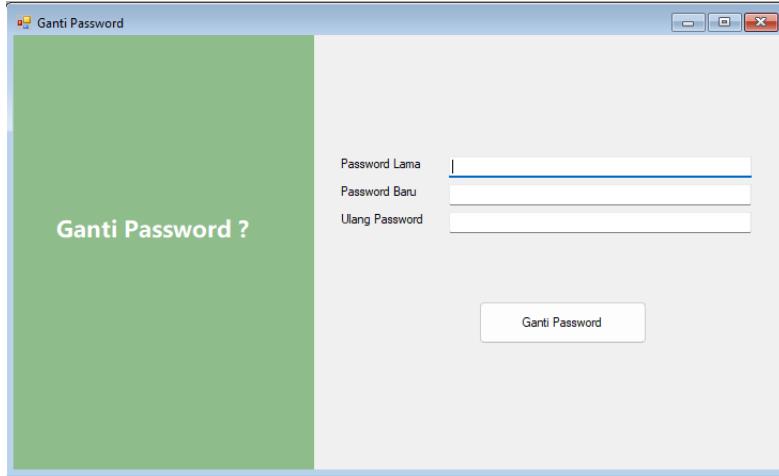
                MessageBox.Show("Koneksi berhasil! Selamat menggunakan aplikasi DiBa!", "Informasi");
                this.DialogResult = DialogResult.OK;
                this.Close();
            }
        }
        else
        {
            FormMain formMain = (FormMain)this.Owner;
            formMain.user = user;
            formMain.saving = Saving.FetchByID(Saving.GenerateID(user));

            MessageBox.Show("Koneksi berhasil! Selamat menggunakan aplikasi DiBa!", "Informasi");
            this.DialogResult = DialogResult.OK;
            this.Close();
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show($"Koneksi gagal! Pesan kesalahan: {exception.Message}", "Error");
    }
}

```

Berikut adalah *code* yang dijalankan ketika pengguna melakukan *login*.

H. Ganti Password



Untuk mengganti password, pengguna dapat melakukannya melalui FormChangePassword.

Ketika mengganti password, pengguna harus memasukkan password lamanya dan memastikan password baru sama dengan password yang diulang.

```
1 reference
private void buttonGanti_Click(object sender, EventArgs e)
{
    try
    {
        FormMain formMain = (FormMain)this.MdiParent;
        User userCheck = User.LoginCheck(formMain.user.Email, textBoxPasswordLama.Text);
        if (userCheck == null)
        {
            throw(new ArgumentException("Password lama salah!"));
        }
        else
        {
            if (textBoxPasswordBaru.Text == textBoxUlangPassword.Text)
            {
                formMain.user.UpdateDate = DateTime.Now;
                User.UpdateData(formMain.user, textBoxPasswordBaru.Text);
                formMain.user = User.FetchDataByID(formMain.user.Nik);
                MessageBox.Show("Password berhasil diubah");
            }
            else
            {
                throw (new ArgumentException("Password yang dimasukkan ulang salah!"));
            }
        }
    }
    catch(Exception exception)
    {
        MessageBox.Show($"Pesan kesalahan: {exception.Message}", "Error");
    }
}
```

Berikut adalah *code* yang dijalankan ketika pengguna mengganti password.

1.3.2 CRUD Jenis Transaksi

A. Class Jenis Transaksi

```
29 references
public class TransactionType
{
    private int idTransType;
    private string code;
    private string name;
```

Untuk memenuhi kebutuhan CRUD tipe transaksi kami membuat *class* baru bernama TrancasctionType. *Class* ini memiliki 3 data member yaitu idTransType , code dan name. Hanya pegawai DiBa saja yang mampu menggunakan *class* ini untuk kebutuhan CRUD.

B. Create Jenis Transaksi

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- AddData
- GenerateID

AddData

```

public static void AddData(TransactionType transactionType)
{
    string command = $"INSERT INTO transaction_types(id_transaction_type, code, " +
        $"name) VALUES({transactionType.IdTransType}, '{transactionType.Code}', '{transactionType.Name}');";
    Connection.ExecuteDML(command);
}

```

Method ini berguna untuk menambah tipe transaksi baru ke dalam database.

```

1 reference
public static int GenerateID()
{
    string command = $"SELECT MAX(id_transaction_type) FROM transaction_types;";
    int hasilID = 0;
    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    if (hasil.Read() == true)
    {
        if (hasil.GetValue(0).ToString() != "")
        {
            int newID = int.Parse(hasil.GetValue(0).ToString()) + 1;
            hasilID = newID;
        }
        else
        {
            hasilID += 1;
        }
    }
    return hasilID;
}

```

GenerateID berfungsi untuk membuat kode id baru untuk tipe transaksi yang akan ditambahkan.

C. Read Jenis Transaksi

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- ReadData
- FetchDataById

ReadData

```

1 reference
public static List<TransactionType> ReadData()
{
    string sql = "SELECT * FROM transaction_types";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    List<TransactionType> listTransactionType = new List<TransactionType>();

    while (hasil.Read() == true)
    {
        TransactionType transactionType = new TransactionType(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
        listTransactionType.Add(transactionType);
    }

    return listTransactionType;
}

1 reference
public static TransactionType FetchDataByID(int idTransactionType)
{
    string sql = $"SELECT * FROM transaction_types WHERE id_transaction_type = {idTransactionType};";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if (hasil.Read() == true)
    {
        TransactionType transactionType = new TransactionType(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
        return transactionType;
    }
    else
    {
        return null;
    }
}

```

Agar program mampu membaca daftar jenis transaksi yang tersimpan di dalam database kami membuat *method baru bernama* ReadData. *Method* ini akan memanggil *method* bernama ExecuteQuerry untuk menjalankan *string* sql berisi perintah untuk mengambil data jenis transaksi dari entitas employees di dalam database.

FetchDataById

```
public static TransactionType FetchDataByID(int idTransactionType)
{
    string sql = $"SELECT * FROM transaction_types WHERE id_transaction_type = {idTransactionType};";
    using (MySqlDataReader hasil = Connection.ExecuteQuery(sql))
    {
        if (hasil.Read() == true)
        {
            TransactionType transactionType = new TransactionType(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
            return transactionType;
        }
        else
        {
            return null;
        }
    }
}
```

Kami juga membuat *method* FetchDataById , yang berfungsi mengambil data jenis transaksi berdasarkan id yang di masukkan.

D. Update Jenis Transaksi

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData

UpdateData

```
1 reference
public static void UpdateData(TransactionType transactionType)
{
    string sql = $"UPDATE transaction_types SET code = '{transactionType.Code}', " +
                $"name = '{transactionType.Name}' WHERE id_transaction_type = {transactionType.IdTransType};";
    Connection.ExecuteDML(sql);
}
```

Untuk bagian *update* , kami membuat *method* baru yaitu UpdateData. Method ini akan menjalankan perintah sql untuk mengubah data yang ada di dalam database.

E. Delete Jenis Transaksi

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- DeleteData

DeleteData

```
1 reference
public static bool DeleteData(TransactionType transactionType)
{
    string command = $"DELETE FROM transaction_types WHERE id_transaction_type = {transactionType.IdTransType};";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Untuk bagian *delete*, kami membuat *method* baru yaitu DeleteData. Method ini akan menjalankan perintah sql untuk menghapus data yang ada di dalam database.

1.3.3 CRUD Position

A. Class Position

```
public class Position
{
    private int idPosition;
    private string name;
    private string description;
```

Untuk memenuhi kebutuhan CRUD posisi kami membuat 1 *class* baru bernama Position. *Class* Position memiliki 3 data member yaitu idPosition , name dan description.

B. Create Position

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

```
1 reference
public static void AddData(Position position)
{
    string command = $"INSERT INTO positions(id_position, name, " +
        $"keterangan) VALUES({position.IdPosition}, '{position.Name}', '{position.Description}');";
    Connection.ExecuteDML(command);
}
```

Untuk bagian *create* , kami membuat 1 *method* bernama AddData. *Method* meminta objek dari *class* Position. Data member dari objek tersebut akan dimasukkan ke dalam variabel command , agar bisa dijalankan oleh *method* ExecuteDML.

C. Read Position

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- ReadData

ReadData

```
3 references
public static List<Position> ReadData()
{
    string sql = "SELECT id_position, name, keterangan FROM positions;";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    List<Position> listPosition = new List<Position>();
    while (hasil.Read() == true)
    {
        Position position = new Position(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
        listPosition.Add(position);
    }
    return listPosition;
}

1 reference
public static Position FetchDataByID(int idPosition)
{
    string sql = $"SELECT * FROM positions WHERE id_position = {idPosition}";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    if (hasil.Read() == true)
    {
        Position position = new Position(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
        return position;
    }
    else
    {
        return null;
    }
}
```

Untuk bagian *read* , agar program mampu membaca daftar jenis posisi yang tersimpan di dalam database kami membuat *method baru bernama* ReadData. *Method* ini akan memanggil *method* bernama ExecuteQuerry untuk menjalankan *string* sql yang berisi perintah untuk mengambil data jenis posisi dari entitas positions di dalam database.

FetchDataById

```
public static Position FetchDataByID(int idPosition)
{
    string sql = $"SELECT * FROM positions WHERE id_position = {idPosition}";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    if (hasil.Read() == true)
    {
        Position position = new Position(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString());
        return position;
    }
    else
    {
        return null;
    }
}
```

Kami juga membuat *method* FetchDataById , yang berfungsi mengambil data jenis posisi berdasarkan id yang di masukkan.

D. Update Position

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- UpdateData

Update

```
1 reference
public static void UpdateData(Position position)
{
    string sql = $"UPDATE positions SET name = '{position.Name.Replace("'", "\\'")}', keterangan " +
        $"= '{position.Description.Replace("'", "\\'")}' WHERE id_position = {position.IdPosition};";
    Connection.ExecuteDML(sql);
}
```

Untuk bagian *update*, kami membuat *method* baru yaitu *UpdateData*. Method ini akan menjalankan perintah sql untuk mengubah data yang ada di dalam database.

E. Delete Position

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- *DeleteData*

DeleteData

```
1 reference
public static bool DeleteData(Position position)
{
    string command = $"DELETE FROM positions WHERE id_position = {position.IdPosition};";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Untuk bagian *delete*, kami membuat *method* baru yaitu *DeleteData*. Method ini akan menjalankan perintah sql untuk menghapus data yang ada di dalam database.

2 Milestone Tahap 2

2.1 CRUD

2.1.1 CRUD Employee

A. Class Employee

```
public class Employee
{
    private int idEmployee;
    private string firstName;
    private string lastName;
    private string nik;
    private string email;
    private string password;
    private DateTime createDate;
    private DateTime updateDate;
    private Position position;
```

Untuk memenuhi kebutuhan CRUD employee kami membuat satu class bernama employee. Class employee memiliki 9 data member.

B. Create Employee

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData
- GenerateId

AddData

```
public static void AddData(Employee employee)
{
    string command = $"INSERT INTO employees(id_employee, first_name, last_name, nik, email, password, create_date, update_date, id_position) VALUES" +
        $" ({employee.IdEmployee}, '{employee.FirstName}', '{employee.LastName}', '{employee.Nik}', '{employee.Email}', SHA2('{employee.Password}', 512), " +
        $" '{employee.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", ""))}', " +
        $"'{employee.UpdateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", ""))}', " +
        $" {employee.Position.IdPosition})";
    Connection.ExecuteDML(command);
}
```

Untuk bagian *Create*, agar program mampu membuat objek baru dari *class Employee* kami membuat satu *method* baru bernama AddData. AddData akan memanggil *method* bernama ExecuteDML dari *class Connection* untuk menjalankan perintah DML agar data pegawai bisa di

C. Read Employee

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- ReadData
- GenerateId
- FetchDataById
- LoginCheck

ReadData

```
public static List<Employee> ReadData()
{
    string sql = "SELECT * FROM employees as e INNER JOIN positions as p ON e.id_position = p.id_position";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    List<Employee> listEmployee = new List<Employee>();

    while (hasil.Read() == true)
    {
        Position position = new Position(int.Parse(hasil.GetValue(9).ToString()), hasil.GetValue(10).ToString(), hasil.GetValue(11).ToString());
        Employee employee = new Employee(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(),
            hasil.GetValue(2).ToString(), hasil.GetValue(3).ToString(), hasil.GetValue(4).ToString(), hasil.GetValue(5).ToString(),
            (DateTime)hasil.GetValue(6), (DateTime)hasil.GetValue(7), position);
        listEmployee.Add(employee);
    }

    return listEmployee;
}
```

Untuk bagian *read* , agar program mampu membaca daftar pegawai yang tersimpan di dalam database kami membuat *method baru bernama* ReadData. *Method* ini akan memanggil *method* bernama ExecuteQuerry untuk menjalankan *string sql* yang berisi perintah untuk mengambil data pegawai dari entitas employees di dalam database.

GenerateID

```
public static int GenerateID()
{
    string command = $"SELECT MAX(id_employee) FROM employees;";
    int hasilID = 0;
    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    if (hasil.Read() == true)
    {
        if (hasil.GetValue(0).ToString() != "")
        {
            int newID = int.Parse(hasil.GetValue(0).ToString()) + 1;
            hasilID = newID;
        }
        else
        {
            hasilID += 1;
        }
    }
    return hasilID;
}
```

Untuk membuat id pegawai secara otomatis , kami membuat *method* bernama GenerateID. Method ini akan mengambil data id terbesar dari entitas employees. Lalu angka terbesar dari id yang didapat akan ditambah 1 dan akan menjadi id baru dari pegawai.

FetchDataById

```
2 references
public static Employee FetchDataByID(int idEmployee)
{
    string sql = $"SELECT * FROM employees as e INNER JOIN positions as p ON e.id_position = p.id_position WHERE e.id_employee = {idEmployee};";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if (hasil.Read() == true)
    {
        Position position = new Position(int.Parse(hasil.GetValue(9).ToString()), hasil.GetValue(10).ToString(), hasil.GetValue(11).ToString());
        Employee employee = new Employee(int.Parse(hasil.GetValue(0).ToString()),
                                         hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString(), hasil.GetValue(3).ToString(), hasil.GetValue(4).ToString(),
                                         hasil.GetValue(5).ToString(), (DateTime)hasil.GetValue(6), (DateTime)hasil.GetValue(7), position);
        return employee;
    }
    else
    {
        return null;
    }
}
```

Kami juga membuat method baru bernama FetchDataByID yang berfungsi untuk mengambil data pegawai berdasarkan id yang dimasukkan ke dalam *argument method*.

LoginCheck

```
1 reference
public static Employee LoginCheck(string email, string password)
{
    string sql = $"SELECT * FROM employees WHERE email = '{email}' AND password = SHA2('{password}', 512);";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    while (hasil.Read() == true)
    {
        Employee employee = Employee.FetchDataByID((int)hasil.GetValue(0));
        return employee;
    }
    return null;
}
```

Method bernama LoginCheck juga kami gunakan untuk mengecek alamat email dan password yang diinput oleh pegawai sudah benar atau belum.

D. Update Employee

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- UpdateData

UpdateData

```
1 reference
public static void UpdateData(Employee employee)
{
    string sql = $"UPDATE employees SET first_name = '{employee.FirstName}', " +
    $"last_name = '{employee.LastName}', nik = '{employee.Nik}', email = '{employee.Email}', " +
    $"password = SHA2('{employee.Password}', 512), create_date = '{employee.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", ""))}', " +
    $"update_date = '{employee.UpdateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", ""))}', " +
    $"id_position = {employee.Position.IdPosition} WHERE id_employee = {employee.IdEmployee};";
    Connection.ExecuteDML(sql);
}
```

Method UpdateData kami gunakan untuk mengubah data pegawai yang tersimpan di dalam database.

E. Delete Employee

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

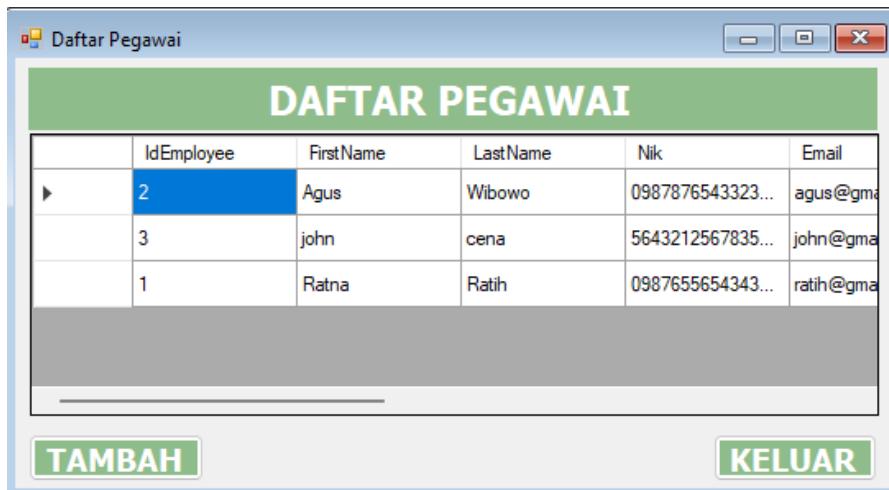
- DeleteData

DeleteData

```
1 reference
public static bool DeleteData(Employee employee)
{
    string command = $"DELETE FROM employees WHERE id_employee = {employee.IdEmployee};";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Untuk bagian *delete*, kami membuat *method* baru yaitu DeleteData. Method ini akan menjalankan perintah sql untuk menghapus data yang ada di dalam database.

F. FormDaftarEmployee



Ketika *form* ini dipanggil, program akan memunculkan daftar pegawai yang disimpan didalam database.

```
4 references
public void FormDaftarPegawai_Load(object sender, EventArgs e)
{
    ListOfEmployees = Employee.ReadData();
    if (ListOfEmployees.Count > 0)
    {
        dataGridViewPegawai.DataSource = listOfEmployees;

        if (dataGridViewPegawai.ColumnCount < 10)
        {
            DataGridViewButtonColumn buttonColumn = new DataGridViewButtonColumn();
            buttonColumn.HeaderText = "Aksi";
            buttonColumn.Text = "Ubah";
            buttonColumn.Name = "btnUbahGrid";
            buttonColumn.UseColumnTextForButtonValue = true;
            dataGridViewPegawai.Columns.Add(buttonColumn);

            DataGridViewButtonColumn buttonColumn1 = new DataGridViewButtonColumn();
            buttonColumn1.HeaderText = "Aksi";
            buttonColumn1.Text = "Hapus";
            buttonColumn1.Name = "btnHapusGrid";
            buttonColumn1.UseColumnTextForButtonValue = true;
            dataGridViewPegawai.Columns.Add(buttonColumn1);
        }
    }
    else
    {
        dataGridViewPegawai.DataSource = null;
    }
}
```

Berikut adalah *code* yang dijalankan ketika FormDaftarPegawai dipanggil

G. FormTambahEmployee

The screenshot shows a Windows application window titled "Tambah Pegawai". The main title bar is green with the text "Tambah Pegawai". The form contains the following fields:

- Kode Pegawai: Input field containing "4".
- Nama Depan: Input field containing "Son".
- Nama Belakang: Input field containing "Goku".
- NIK: Input field containing "123124124141313".
- Email Pegawai: Input field containing "son@gmail.com".
- Password: Input field containing "***".
- Posisi: A dropdown menu currently set to "Admin".
- Buttons at the bottom: "Simpan" (Save) and "Keluar" (Exit).

Untuk menambah pegawai akan melalui FormTambahEmployee

The screenshot shows a Windows application window titled "Daftar Pegawai". The main title bar is blue with the text "DAFTAR PEGAWAI". The table displays the following data:

	IdEmployee	FirstName	LastName	Nik	Email
▶	2	Agus	Wibowo	0987876543323...	agus@gma...
	3	john	cena	5643212567835...	john@gma...
	4	Son	Goku	123124124141313	son@gmai...
	1	Ratna	Ratih	0987655654343...	ratih@gma...

Buttons at the bottom: "TAMBAH" (Add) and "KELUAR" (Exit).

Setelah proses menambah pegawai berhasil, data pegawai baru akan muncul di FormDaftarPegawai

2.1.2 CRUD Inbox

A. Class Inbox

```
25 references
public class Inbox
{
    private int idMessage;
    private string messageContent;
    private DateTime sendDate;
    private string status;
    private DateTime updateDate;
    private User user;
```

Untuk memenuhi kebutuhan CRUD dari inbox kami membuat *class* baru bernama *Inbox*. *Class* ini memiliki 6 data member.

B. Create Inbox

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

```
1 reference
public static void AddData(Inbox message)
{
    string command = $"INSERT INTO inboxs(id_message, message_content," +
        $" send_date, status, update_date, users_nik) VALUES ({message.IdMessage}, " +
        $" '{message.MessageContent}', '{message.SendDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $" '{message.Status}', '{message.UpdateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', '{message.User.Nik}');";
    Connection.ExecuteNonQuery(command);
}
```

Untuk bagian *Create*, agar program mampu membuat objek baru dari *class* *Inbox* kami membuat satu *method* baru bernama *AddData*. *AddData* akan memanggil *method* bernama *ExecuteDML* dari *class* *Connection* untuk menjalankan perintah DML agar data pegawai bisa dimasukkan ke dalam database.

C. Read Inbox

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- ReadData
- FetchDataById
- GenerateId

ReadData

```
2 references
public static List<Inbox> ReadData(string nik)
{
    string command;
    if (nik == "")
    {
        command = $"SELECT * FROM inboxs;";
    }
    else
    {
        command = $"SELECT * FROM inboxs WHERE users_nik = '{nik}'";
    }

    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    List<Inbox> listOfMessage = new List<Inbox>();
    while (hasil.Read())
    {
        User user = User.FetchDataByID(hasil.GetString(5));
        Inbox message = new Inbox((int)hasil.GetValue(0), hasil.GetString(1), (DateTime)hasil.GetValue(2), hasil.GetString(3), (DateTime)hasil.GetValue(4), user);
        listOfMessage.Add(message);
    }
    return listOfMessage;
}
```

Untuk bagian *read* , agar program mampu membaca daftar inbox yang tersimpan di dalam database kami membuat *method baru bernama* ReadData. *Method* ini akan memanggil *method* bernama ExecuteQuerry untuk menjalankan *string sql* yang berisi perintah untuk mengambil data inbox dan user dari entitas employees dan users di dalam database.

FetchDataById

```
1 reference
public static Inbox FetchDataByID(int id, string nik)
{
    string command = $"SELECT * FROM inboxs WHERE id_message = {id} AND users_nik = '{nik}'";
    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    if (hasil.Read() == true)
    {
        User user = User.FetchDataByID(hasil.GetString(5));
        Inbox message = new Inbox((int)hasil.GetValue(0), hasil.GetString(1), (DateTime)hasil.GetValue(2), hasil.GetString(3), (DateTime)hasil.GetValue(4), user);
        return message;
    }
    else
    {
        return null;
    }
}
```

Kami juga membuat method baru bernama FetchDataByID yang berfungsi untuk mengambil data inbox berdasarkan id inbox dan id user pemilik inbox yang dimasukkan ke dalam parameter *method*.

GenerateID

```
1 reference
public static int GenerateID()
{
    string command = $"SELECT MAX(id_message) FROM inboxs";
    int hasilID = 0;
    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    if (hasil.Read() == true)
    {
        if (hasil.GetValue(0).ToString() != "")
        {
            int newID = int.Parse(hasil.GetValue(0).ToString()) + 1;
            hasilID = newID;
        }
        else
        {
            hasilID += 1;
        }
    }
    return hasilID;
}
```

Untuk membuat id pegawai secara otomatis , kami membuat *method* bernama GenerateID. Method ini akan mengambil data id terbesar dari entitas inboxs. Lalu angka terbesar dari id yang didapat akan ditambah 1 dan akan menjadi id baru dari inbox.

D. Update Inbox

```
!reference
public static void UpdateData(Inbox message)
{
    string command = $"UPDATE inboxs SET message_content " +
        $"= '{message.MessageContent}', send_date = '{message.SendDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}' , " +
        $" status = '{message.Status}', update_date = '{message.UpdateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}' ;" +
        $"WHERE id_message = {message.IdMessage} AND users_nik = '{message.User.Nik}' ;";
    Connection.ExecuteDML(command);
}
```

Method UpdateData kami gunakan untuk mengubah data inbox yang tersimpan di dalam database.

E. Delete Inbox

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

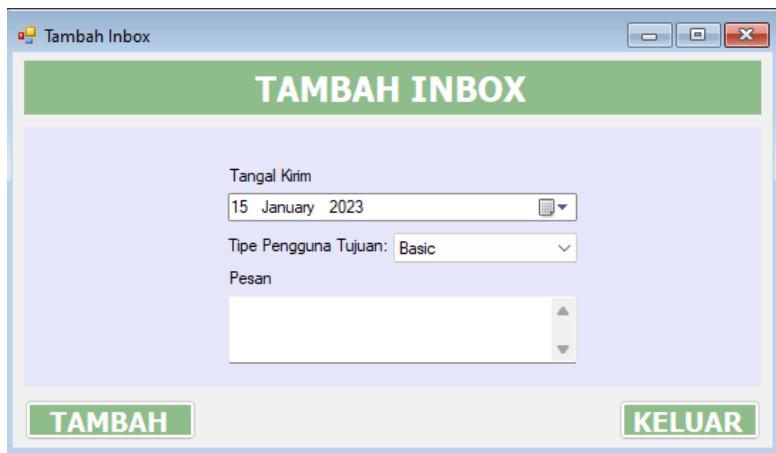
- DeleteData

DeleteData

```
!reference
public static bool DeleteData(Inbox message)
{
    string command = $"DELETE FROM inboxs WHERE id_message = {message.IdMessage} AND users_nik = '{message.User.Nik}' ;";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Untuk bagian *delete*, kami membuat *method* baru yaitu DeleteData. Method ini akan menjalankan perintah sql untuk menghapus data yang ada di dalam database.

F. FormTambahInbox

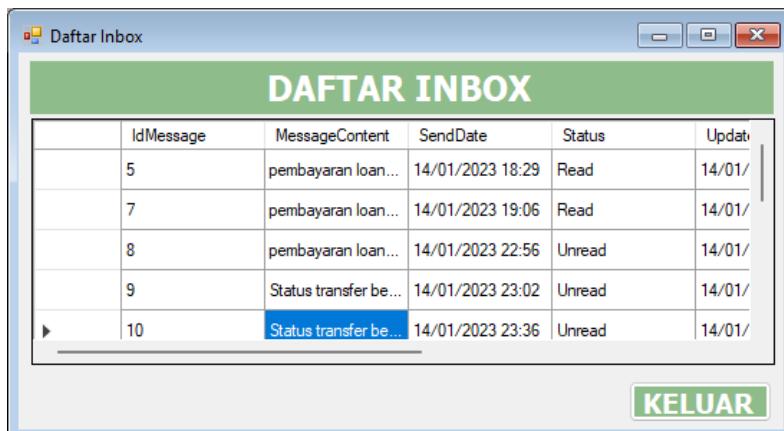


Hanya pegawai DiBa yang memiliki akses untuk membuat pengumuman kepada user. Pegawai dapat memilih tipe user apa yang akan mendapatkan pengumuman yang telah dibuat.

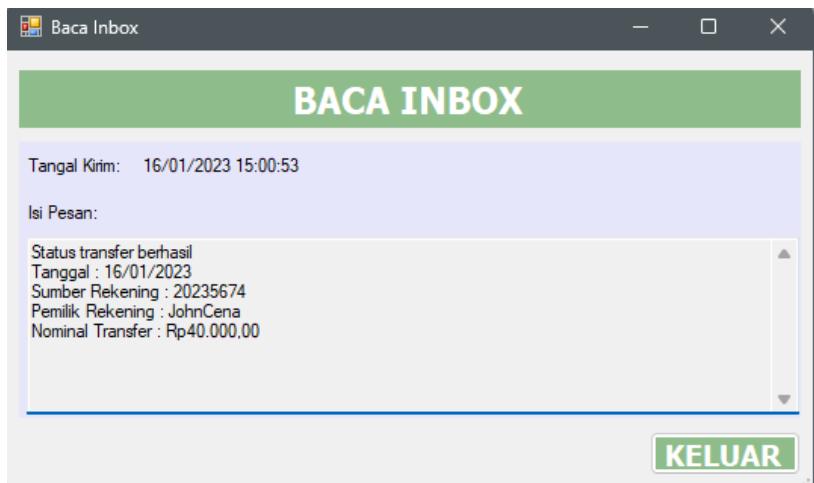


G. FormDaftarInbox

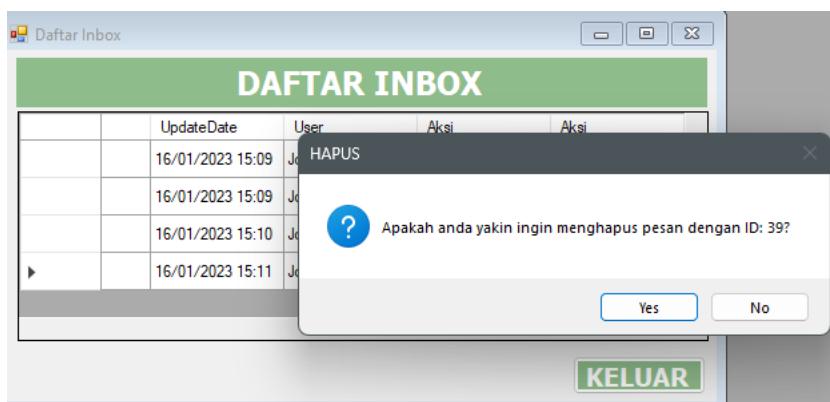
Pengguna dapat melihat daftar inbox yang berisi pesan transaksi ataupun pengumuman dari DiBa melalui FormDaftarInbox.



Ketika pengguna menekan button baca, maka pengguna akan membaca pesan inbox secara lebih detail.



Jika pengguna menekan *button* hapus, maka pengguna dapat menghapus pesan yang dipilih



```
public void FormDaftarInbox_Load(object sender, EventArgs e)
{
    FormMain FormMain = (FormMain)this.MdiParent;
    if (FormMain.employee != null)
    {
        listOfMessage = Inbox.ReadData("");
    }
    else
    {
        listOfMessage = Inbox.ReadData(FormMain.user.Nik);
    }

    if (listOfMessage.Count > 0)
    {
        dataGridViewInbox.DataSource = listOfMessage;
        if (dataGridViewInbox.ColumnCount < 7)
        {
            DataGridViewButtonColumn buttonColumn = new DataGridViewButtonColumn();
            buttonColumn.HeaderText = "Aksi";
            buttonColumn.Text = "Baca";
            buttonColumn.Name = "btnBacaGrid";
            buttonColumn.UseColumnTextForButtonValue = true;
            dataGridViewInbox.Columns.Add(buttonColumn);

            DataGridViewButtonColumn buttonColumn1 = new DataGridViewButtonColumn();
            buttonColumn1.HeaderText = "Aksi";
            buttonColumn1.Text = "Hapus";
            buttonColumn1.Name = "btnHapusGrid";
            buttonColumn1.UseColumnTextForButtonValue = true;
            dataGridViewInbox.Columns.Add(buttonColumn1);
        }
    }
    else
    {
        dataGridViewInbox.DataSource = null;
    }
}
```

Berikut adalah *code* dari FormDaftarInbox ketika di *load*.

```

if (e.ColumnIndex == dataGridViewInbox.Columns["btnHapusGrid"].Index && e.RowIndex >= 0)
{
    try
    {
        int deleteIdMessage = int.Parse(dataGridViewInbox.CurrentRow.Cells["IdMessage"].Value.ToString());
        User deletedUser = (User)dataGridViewInbox.CurrentRow.Cells["User"].Value;

        DialogResult hasil = MessageBox.Show($"Apakah anda yakin ingin menghapus pesan dengan ID: {deleteIdMessage}?", "HAPUS", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (hasil == DialogResult.Yes)
        {
            bool hapus = Inbox.DeleteData(inbox.FetchDataByID(deleteIdMessage, deletedUser.NIK));
            if (hapus == true)
            {
                MessageBox.Show("Penghapusan data berhasil");
                FormDaftarInbox_Load(buttonMeluar, e);
            }
            else
            {
                MessageBox.Show("Penghapusan data gagal");
            }
        }
    }
    catch(Exception exception)
    {
        MessageBox.Show($"*Peser kesalahan: {exception.Message}");
    }
}
else if (e.ColumnIndex == dataGridViewInbox.Columns["btnBacaGrid"].Index && e.RowIndex >= 0)
{
    FormBacaInbox formBacaInbox = new FormBacaInbox();
    formBacaInbox.Owner = this;

    FormBacaInbox.currentMessage.IdMessage = int.Parse(dataGridViewInbox.CurrentRow.Cells["IdMessage"].Value.ToString());
    FormBacaInbox.currentMessage.SendDate = (DateTime)dataGridViewInbox.CurrentRow.Cells["SendDate"].Value;
    FormBacaInbox.currentMessage.Status = dataGridViewInbox.CurrentRow.Cells["Status"].Value.ToString();
    FormBacaInbox.currentMessage.UpdateDate = (DateTime)dataGridViewInbox.CurrentRow.Cells["UpdateDate"].Value;
    FormBacaInbox.currentMessage.User = (User)dataGridViewInbox.CurrentRow.Cells["User"].Value;

    formBacaInbox.ShowDialog();
}
}

```

Berikut adalah *code* dari *button* baca dan *button* hapus di dalam FormDaftarInbox

3 Milestone Tahap 3

3.1 CRUD

3.1.1 CRUD Address Book

A. Class Address Book

```

21 references
public class AddressBook
{
    private User user;
    private Saving saving;
    private string description;
}

```

Untuk memenuhi kebutuhan CRUD dari address Book kami membuat 1 *class* baru bernama “AddressBook”.

B. Create Address Book

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- TambahData

TambahData

```
2 references
public static bool TambahData(string idUser , string accountNumber , string keterangan)
{
    string sql = "insert into addressbook(id_users , account_number , keterangan) " +
        "values('" + idUser + "' , '" + accountNumber + "' , '" + keterangan + "')";
    int hasil = Connection.ExecutedML(sql);

    bool cek;
    if (hasil != 0)
    {
        cek = true;
    }
    else
    {
        cek = false;
    }
    return cek;
}
```

Method ini berfungsi untuk menambah data kontak rekening ke dalam database

C. Read Address Book

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- BacaData
- FetchDataById

BacaData

```
2 references
public static List<AddressBook> BacaData(string kriteria, string nilaiKriteria, User user)
{
    string sql;
    if (kriteria == "")
    {
        sql = "select * from addressbook " +
            "where id_users = '" + user.Nik + "'";
    }
    else
    {
        sql = "select * from addressbook " +
            "where id_users = '" + user.Nik + "' " +
            "and " + kriteria + " like '%" + nilaiKriteria + "%'";
    }

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    List<AddressBook> listAddress = new List<AddressBook>();
    while (hasil.Read() == true)
    {
        User pengguna = User.FetchDataByID(hasil.GetValue(0).ToString());
        Saving saving = Saving.FetchByDataID(hasil.GetValue(1).ToString());
        AddressBook adBook = new AddressBook(pengguna, saving, hasil.GetValue(2).ToString());

        listAddress.Add(adBook);
    }

    return listAddress;
}
```

Method ini akan mengambil data kontak rekening yang dimiliki oleh pengguna yang *login* ke dalam DiBa.

FetchDataById

```
3 references
public static AddressBook FetchDataByID(Saving saving , User user)
{
    string sql = "select * from addressbook " +
        "where id_users like '%" + user.Nik + "%' " +
        "and account_number like '%" + saving.AccountNumber + "%' ";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if(hasil.Read() == true)
    {
        User pengguna = User.FetchByID(hasil.GetValue(0).ToString());
        Saving tabungan = Saving.FetchbyDataID(hasil.GetValue(1).ToString());
        AddressBook adBook = new AddressBook(pengguna, tabungan, hasil.GetValue(2).ToString());
        return adBook;
    }
    else
    {
        return null;
    }
}
```

Method ini akan mengambil 1 data kontak rekening saja.

D. Update Address Book

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData

UpdateData

```
1 reference
public static bool UpdateData(string updateKeterangan, string noRek , User user)
{
    string sql = "update addressbook " +
        "set keterangan = '" + updateKeterangan + "' " +
        "where id_users ='" + user.Nik + "' " +
        "and account_number = '" + noRek + "'";

    int hasil = Connection.ExecuteDML(sql);
    bool cek;
    if(hasil != 0)
    {
        cek = true;
    }
    else
    {
        cek = false;
    }
    return cek;
}
```

Method ini berfungsi untuk mengubah data kontak rekening yang ada di database.

E. Delete Address Book

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- DeleteData

DeleteData

```
1 reference
public static bool DeleteData(string noRek,User user)
{
    string sql = "Delete from addressbook "
        "where id_users like '%" + user.Nik + "%' "
        "and account_number like '%" + noRek + "%' ";
    int hasil = Connection.ExecuteNonQuery(sql);

    bool cek;
    if (hasil != 0)
    {
        cek = true;
    }
    else
    {
        cek = false;
    }
    return cek;
}
```

Method ini berfungsi untuk menghapus data kontak rekening yang ada di database.

F. FormAddressBook



Ketika *form* ini dipanggil, maka program akan menjalankan *method* ReadData untuk mengambil data kontak rekening yang dimiliki oleh pengguna.

```

public void FormAddressBook_Load(object sender, EventArgs e)
{
    FormMain frmMenu = (FormMain)this.MdiParent;
    user = frmMenu.user;

    listAddressBook = AddressBook.BacaData("", "", frmMenu.user);
    tabunganSendiri = frmMenu.saving;

    FormatDataGridView();
    dataGridViewAddressBook.Rows.Clear();
    TampilDataGridView();
}

```

Berikut adalah *code* yang dijalankan ketika FormAddressBook di *load*.

G. FormTambahAddressBook



FormTambahAddressBook akan digunakan ketika pengguna ingin menambah kontak rekening. Ketika pengguna ingin menambah rekening, maka program akan mengecek apakah kontak rekening yang ingin ditambah sudah ada di dalam daftar kontak atau belum jika sudah maka program akan memberi informasi kepada pengguna bahwa nomor rekening tersebut sudah ada di dalam daftar kontak.



Jika kontak rekening yang ingin ditambah, adalah rekening yang sudah ditutup. Maka program akan memberi informasi kepada pengguna bahwa nomor rekening tersebut sudah ditutup.



Jika penambahan kontak rekening berhasil, maka nomor rekening yang baru ditambah akan muncul secara otomatis di dalam daftar kontak rekening.



DAFTAR REKENING				
No Rekening	Nama	Keterangan	ubah	hapus
20222345	Siryo		Ubah	Hapus
20223456	Supra	Pengutang abadi	Ubah	Hapus

```

    T Reference
private void buttonTambah_Click(object sender, EventArgs e)
{
    try
    {
        if (textBoxNoRekening.Text == tabungan.AccountNumber)
        {
            throw(new ArgumentException("tidak bisa memasukkan tabungan sendiri !"));
        }
        else
        {
            Saving cekRekening = Saving.FetchByID(textBoxNoRekening.Text);
            if (cekRekening == null)
            {
                throw (new ArgumentException("kontak rekening tidak ditemukan :("));
            }
            else if(cekRekening.Status == "Closed")
            {
                throw (new ArgumentException("Mohon maaf rekening ini sudah ditutup"));
            }
            else
            {
                AddressBook ab = AddressBook.FetchDataByID(cekRekening, user);
                if(ab != null)
                {
                    throw (new ArgumentException("Rekening ini sudah terdaftar di address book"));
                }
                else
                {
                    bool cek = AddressBook.TambahData(user.Nik, cekRekening.AccountNumber, textBoxKeterangan.Text);
                    if (cek == true)
                    {
                        MessageBox.Show("tambah kontak rekening berhasil !");
                    }
                    else
                    {
                        throw (new ArgumentException("tambah kontak rekening gagal :("));
                    }
                }
            }
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show("pesan kesalahan : " + ex.Message);
    }
}

```

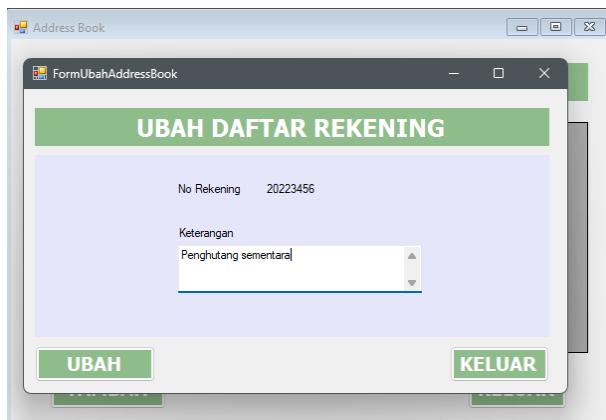
Berikut adalah *code* yang dijalankan ketika pengguna menambahkan kontak rekening

H. FormUbahRekening

Form ini akan dipanggil ketika pengguna mengklik kolom ubah di baris kontak rekening yang ingin diubah.



Informasi yang dapat dibuat oleh pengguna adalah deskripsi dari kontak rekening tersebut.



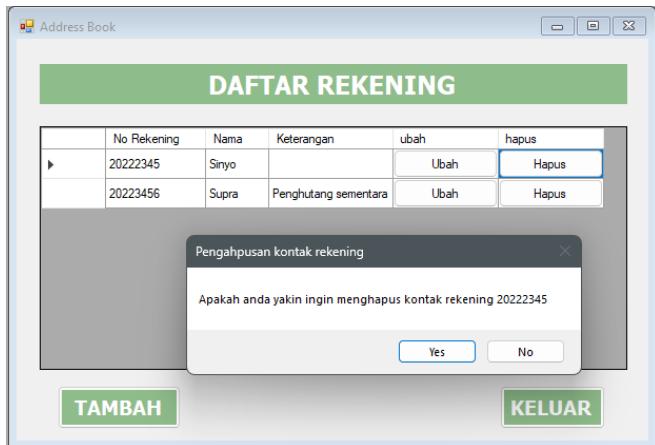
Setelah perubahan berhasil maka, hasil perubahan akan muncul di FormAddressBook



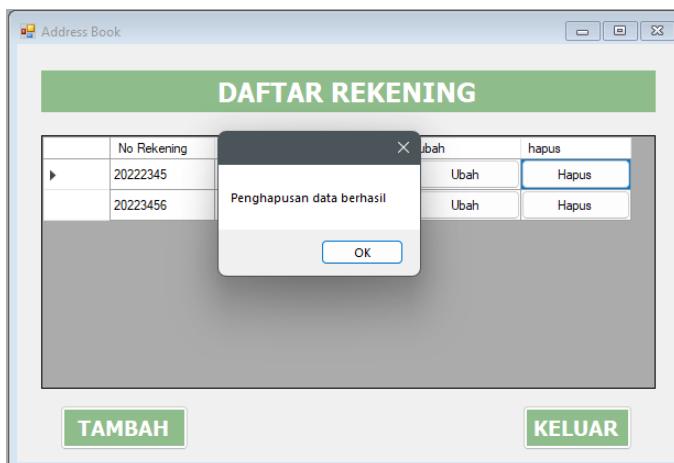
```
try
{
    bool cek = AddressBook.UpdateData(textBoxKeterangan.Text, labelNoRek.Text, pengguna);
    if(cek == true)
    {
        MessageBox.Show("Ubah data berhasil !");
    }
    else
    {
        MessageBox.Show("Ubah data gagal !");
    }
}
catch(Exception ex)
{
    MessageBox.Show("Pesan kesalahan " + ex.Message);
}
```

Berikut adalah *code* yang dijalankan ketika pengguna mengubah keterangan kontak rekening.

I. Hapus Kontak Rekening



Untuk menghapus kontak rekening pengguna cukup mengklik kolom hapus di baris kontak rekening yang ingin dihapus.



```
if(e.ColumnIndex == dataGridViewAddressBook.Columns["buttonHapus"].Index && e.RowIndex >= 0)
{
    try
    {
        string noRekening = dataGridViewAddressBook.CurrentRow.Cells["norekening"].Value.ToString();
        DialogResult hasil = MessageBox.Show("Apakah anda yakin ingin menghapus kontak rekening " +
            noRekening, "Penghapusan kontak rekening", MessageBoxButtons.YesNo);

        if(hasil == DialogResult.Yes)
        {
            bool cek = AddressBook.DeleteData(noRekening, user);
            if(cek == true)
            {
                MessageBox.Show("Penghapusan data berhasil");
                FormAddressBook_Load(buttonKeluar, e);
            }
            else
            {
                MessageBox.Show("penghapusan data gagal");
            }
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show("pesan kesalahan = " + ex.Message);
    }
}
```

Berikut adalah *code* yang dijalankan ketika pengguna menghapus kontak rekening.

3.1.2 CRUD Transaksi

A. Class Transaksi

```
1 reference
public class Transaction
{
    private string idTransaction;
    private DateTime transactionDate;
    private double nominal;
    private string description;
    private Saving sourceAccount;
    private Saving destinationAccount;
    private TransactionType type;
    private Merchant merchant;

    public Transaction(string idTransaction, DateTime transactionDate, double nominal, string description, Saving sourceAccount, Saving destinationAccount, TransactionType type, Merchant merchant)
    {
        IdTransaction = idTransaction;
        TransactionDate = transactionDate;
        Nominal = nominal;
        Description = description;
        SourceAccount = sourceAccount;
        DestinationAccount = destinationAccount;
        Type = type;
        Merchant = pMerchant;
    }
}
```

Untuk memenuhi kebutuhan transaksi kami membuat *class* transaction.

B. Create Transaksi

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- TambahTransaksi
- TopUp

TambahTransaksi

```
1 reference
public static void TambahTransaksi(Transaction transaksi)
{
    using (TransactionScope transcope = new TransactionScope())
    {
        try
        {
            Connection koneksi = new Connection();
            Connection.ExecuteNonQuery(transaksi.GenerateCommandSql(transaksi), koneksi);
            UpdateSavingSend(transaksi.Nominal, transaksi.SourceAccount, koneksi);
            UpdateSavingReceive(transaksi.Nominal, transaksi.DestinationAccount, koneksi);
            transcope.Complete();
        }
        catch(Exception ex)
        {
            throw new Exception(ex.Message);
        }
    }
}
```

Untuk membuat melakukan create transaksi dilakukan dengan menggunakan *method* TambahTransaksi yang ada di dalam *class* transaction. *Method* TambahTransaksi juga akan menggunakan transcope untuk memastikan semua perintah DML berhasil dijalankan.

Kami juga menggunakan konsep *method overloading* untuk membuat varian *method* TambahTransaksi untuk deposito dan hutang,

```

2 references
public static void TambahTransaksi(Transaction transaksi , Deposit deposito ,string tipe)
{
    using (TransactionScope transcope = new TransactionScope())
    {
        try
        {
            if (tipe == "konfirmDeposit")//mengurangi saldo tabungan ketika konfirm deposit
            {
                Connection koneksi = new Connection();
                Connection.ExecuteNonQuery(GenerateCommandSql(transaksi), koneksi);
                UpdateSavingSend(deposito.Nominal, deposito.Saving.AccountNumber, koneksi);
                transcope.Complete();
            }
            else
            {
                Connection koneksi = new Connection(); //menambah saldo tabungan ketika tutup deposit
                Connection.ExecuteNonQuery(GenerateCommandSql(transaksi), koneksi);
                UpdateSavingReceive(deposito.Nominal, deposito.Saving.AccountNumber, koneksi);
                transcope.Complete();
            }
        }
        catch (Exception ex)
        {
            transcope.Dispose();
            throw new Exception(ex.Message);
        }
    }
}

```

Berikut adalah *method* TambahTransaksi untuk pengajuan deposito dan pencairan deposito.

```

2 references
public static void TambahTransaksi(Transaction transaksi , Loan loan , string tipe)//method dipanggil untuk proses
//pengajuan hutang dan pembayaran hutang
{
    using (TransactionScope transcope = new TransactionScope())
    {
        try
        {
            if (tipe == "konfirmLoan")//ketika hutang dikonfirmasi oleh petugas diba, maka program akan otomatis mengurangi saldo
                //dari tabungan milik aplikasi Diba
                //dan tabungan dari si pembuat hutang akan
                //otomatis mendapatkan saldo tambahan senilai nominal hutang
            {
                Connection koneksi = new Connection();
                Connection.ExecuteNonQuery(GenerateCommandSql(transaksi), koneksi);
                UpdateSavingSend(loan.Amount ,transaksi.SourceAccount.AccountNumber, koneksi);
                UpdateSavingReceive(loan.Amount ,loan.Lender.AccountNumber, koneksi);
                transScope.Complete();
            }
            else
            {
                //ketika pengguna membayar hutang, maka otomatis saldo pengguna berkurang
                //senilai nominal hutang yang dibayar
                //dan saldo dari tabungan milik aplikasi Diba akan bertambah senilai,
                //nominal hutang yang dibayar.
                Connection koneksi = new Connection();
                Connection.ExecuteNonQuery(GenerateCommandSql(transaksi), koneksi);
                UpdateSavingSend(loan.Amount , loan.Lender.AccountNumber, koneksi);
                UpdateSavingReceive(loan.Amount ,transaksi.DestinationAccount.AccountNumber, koneksi);
                transScope.Complete();
            }
        }
        catch(Exception ex)
        {
            transScope.Dispose();
            throw (new ArgumentException(ex.Message));
        }
    }
}

```

Berikut adalah *method* TambahTransaksi untuk pengajuan hutang dan pembayaran hutang.

TopUp

```
1 reference
public static void TopUP(Transaction transaksi)
{
    using (TransactionScope transcope = new TransactionScope())
    {
        try
        {
            Connection koneksi = new Connection();
            Connection.ExecuteDMLTranscope(GenerateCommandSql(transaksi), koneksi);
            string sql = "update savings " +
                        "set balance = balance + " + transaksi.Nominal + " " +
                        "where account_number = '" + transaksi.DestinationAccount.AccountNumber + "' ";
            Connection.ExecuteDMLTranscope(sql, koneksi);
            transcope.Complete();
        }
        catch(Exception ex)
        {
            transcope.Dispose();
            throw (new ArgumentException(ex.Message));
        }
    }
}
```

Terdapat juga *method* TopUp , *method* ini akan membuat pengguna mampu melakukan top up saldo ke tabungan yang dimiliki. Kami mengasumsi bahwa asal tabungan dan tujuan tabungan di dalam transaksi TopUp adalah sama , karena tujuan dari TopUp saldo adalah tabungan pengguna yang login.

C. Read Transaksi

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- BacaData

ReadData

```
0 references
public static List<Transaction> BacaData(string kriteria, string nilaiKriteria, Saving asal)
{
    string sql;
    if (kriteria == "")
    {
        sql = "select * from transactions " +
              "where source_account_number = '" + asal.AccountNumber + "' ";
    }
    else
    {
        sql = "select * from transactions " +
              "where source_account_number = '" + asal.AccountNumber + "' " +
              "and " + kriteria + " like '%" + nilaiKriteria + "%'";
    }

    MySqlDataReader hasil = Connection.ExecuteNonQuery(sql);
    List<Transaction> listTransaction = new List<Transaction>();
    while(hasil.Read()==true)
    {
        Merchant merc = Merchant.FetchDataByID.Parse(hasil.GetValue(7).ToString());
        TransactionType tipe = TransactionType.FetchDataByID.Parse(hasil.GetValue(6).ToString());
        Saving destination = Saving.FetchByID(hasil.GetValue(5).ToString());
        Saving source = Saving.FetchByID(hasil.GetValue(4).ToString());
        Transaction transaction = new Transaction(hasil.GetValue(0).ToString(), DateTime.Parse(hasil.GetValue(1).ToString()),
                                                    Double.Parse(hasil.GetValue(2).ToString()), hasil.GetValue(3).ToString(), source, destination, tipe, merc);
        listTransaction.Add(transaction);
    }
    return listTransaction;
}
```

Method BacaData di dalam *class* transaction berguna untuk melakukan read entitas transaksi di dalam database.

D. Update Transaksi

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateTransaction

UpdateTransaction

```
0 references
public static void UpdateTransaction(Transaction transaction)
{
    string command = $"UPDATE FROM transactions SET transaction_date = " +
        $" '{transaction.TransactionDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $" nominal = {transaction.Nominal}, keterangan = '{transaction.Description}', id_transaction_type = {transaction.Type.IdTransType}, " +
        $" merchant_id = {transaction.Merchant.Id} WHERE id_transaction = {transaction.IdTransaction} AND source_account_number = '{transaction.SourceAccount.AccountNumber}'";
    Connection.ExecuteNonQuery(command);
}
```

Method UpdateTransaction di dalam *class* transaction berguna untuk mengubah data transaksi di dalam database.

E. Delete Transaksi

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

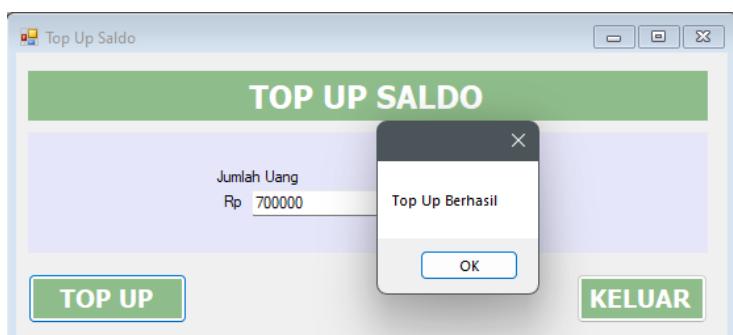
- DeleteTransaction

DeleteTransction

```
0 references
public static bool DeleteTransaction(Transaction transaction)
{
    string command = $"DELETE FROM transactions WHERE id_transaction = '{transaction.IdTransaction}'";
    int rowAffected = Connection.ExecuteNonQuery(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Method DeleteTransaction di dalam *class* transaction berguna untuk menghapus data transaksi di dalam database.

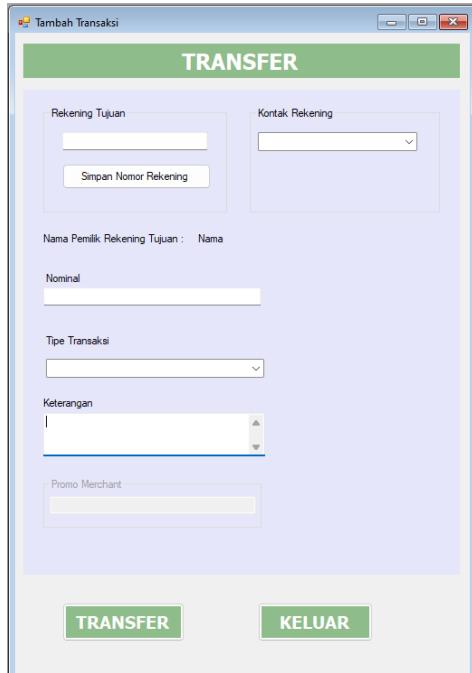
F. FormTopUp



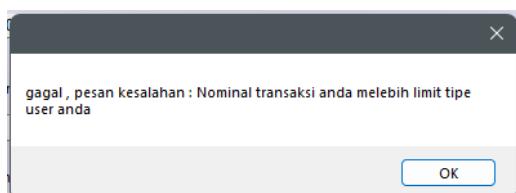
Setelah rekening tabungan pengguna aktif, maka pengguna dapat melakukan *top up* saldo untuk mengisi tabungan mereka.

```
private void buttonTambah_Click(object sender, EventArgs e)
{
    try
    {
        Merchant merc = new Merchant();
        Transaction transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now,
            double.Parse(textBoxNominal.Text), "Top up", tabungan, tabungan,
            TransactionType.FetchDataByID(1), merc);
        Transaction.TopUP(transaksi);
        MessageBox.Show("Top Up Berhasil");
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

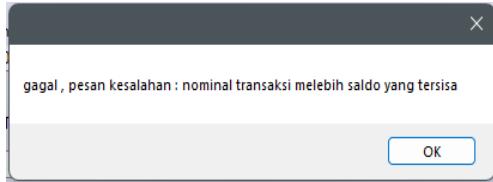
G. FormTambahTransaksi.



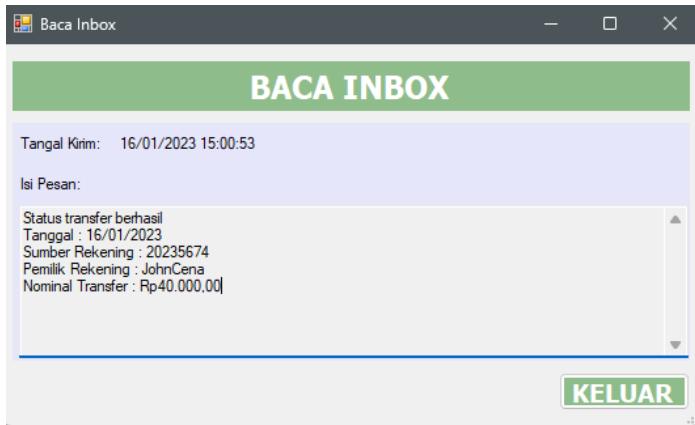
Form ini akan digunakan ketika pengguna ingin melakukan transaksi dengan akun rekening DiBa lainnya. Pengguna dapat menyimpan nomor rekening baru ke dalam daftar kontak rekening atau pengguna juga dapat memilih rekening tujuan mereka berdasarkan daftar kontak rekening yang dimiliki. Jika rekening tujuan adalah rekening dengan tipe *merchant*, maka pengguna dapat memakai promo merchant dalam transaksi mereka. Pengguna dapat memilih 2 tipe transaksi yang digunakan yakni credit atau debit. Ketika pengguna melakukan transaksi, program akan melakukan pengecekan seperti apakah nominal transaksi melebihi *limit* transaksi dari tipe *user* pengguna saat ini dan apakah nominal transaksi melebihi sisa tabungan pengguna. Program juga akan mengecek apakah saldo pengguna kurang dari saldo minimum atau tidak.



Program akan memberi peringatan jika nominal transaksi user melebihi *limit* transaksi.



Program juga akan memberi peringatan jika nominal transaksi user melebihi saldo yang tersisa.



Jika transaksi berhasil program akan mengirim pesan ke inbox pengguna.

```
private void FormTambahTransaksi_Load(object sender, EventArgs e)
{
    frmMain = (FormMain)this.MdiParent;
    pengguna = frmMain.user;
    groupBoxPromoMerchant.Enabled = false;
    asal = frmMain.saving;
    listTipe = TransactionType.ReadData();
    comboBoxTipeTransaksi.Items.Add(listTipe[0]);
    comboBoxTipeTransaksi.Items.Add(listTipe[1]);
    comboBoxTipeTransaksi.DisplayMember = "Nama";
    listAddressBook = AddressBook.BacaData("", "", pengguna);
    foreach(AddressBook ab in listAddressBook)
    {
        comboBoxKontak.Items.Add(ab.Saving.AccountNumber);
    }
}
```

Berikut adalah *code* yang dijalankan ketika FormTambahTransaksi di-*load*.

```

    I reference
private void buttonTambah_Click(object sender, EventArgs e)
{
    try
    {
        if (groupBoxPromoMerchant.Enabled == true)
        {
            mmerc = Merchant.CariPromo(textBoxKodePromo.Text);
            bool cek = Merchant.CekTanggal(mmerc, DateTime.Now);
            if(cek == true)
            {
                cekMerchant = true;
            }
            else
            {
                mmerc = new Merchant();
                cekMerchant = false;
                throw (new ArgumentException("Anda menggunakan promo di luar ketentuan tanggal"));
            }
        }
        else
        {
            mmerc = new Merchant();
            cekMerchant = false;
        }

        Double nominal = double.Parse(textBoxNominal.Text);
        Saving cekSaldo = Saving.FetchByDataID(asal.AccountNumber);

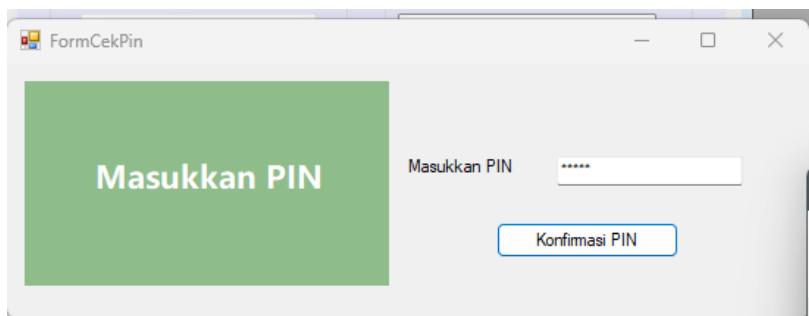
        if(cekSaldo.Balance > 50000)
        {
            if (nominal > cekSaldo.Balance)
            {
                throw (new ArgumentException("nominal transaksi melebihi saldo yang tersisa"));
            }
            else
            {
                if(nominal > pengguna.UserType.TransactionLimit) // melakukan pengecekan apakah nominal transfer melebihi limit transfer dari tipe transaksi
                {
                    throw (new ArgumentException("Nominal transaksi anda melebihi limit tipe user anda"));
                }
                else
                {
                    transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now, nominal, noRekening,
                        asal, tujuan, type, mmerc);

                    FormCekPIN formCekPIN = new FormCekPIN();
                    formCekPIN.Dance = this;
                    formCekPIN.Show();
                }
            }
        }
        else
        {
            throw (new ArgumentException("Saldo tabungan kurang dari minimum sisa saldo , mohon top up terlebih dahulu"));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("gagal , pesan kesalahan : " + ex.Message);
    }
}

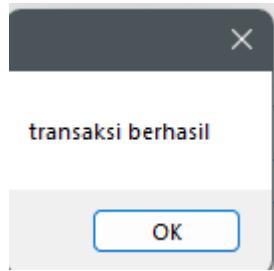
```

Berikut adalah *code* yang djalankan ketika pengguna melakukan transaksi.

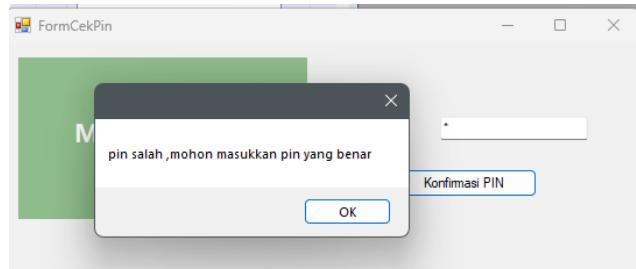
H. FormCekPin



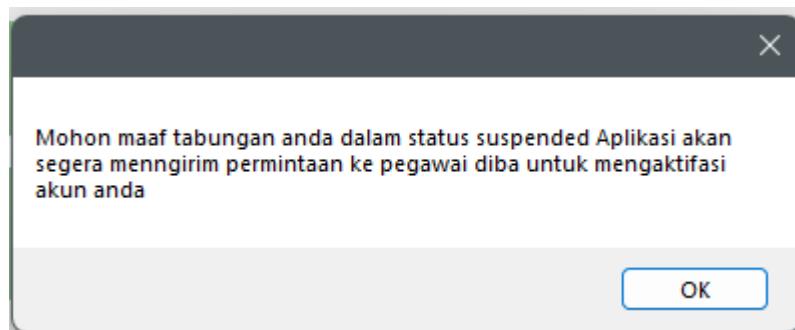
Form ini akan muncul ketika pengguna berhasil melewati pengecekan nominal transaksi. Sesuai dengan namanya, FormCekPin berfungsi untuk melakukan pengecekan PIN pengguna. Jika pengguna berhasil memasukkan pin yang benar maka program akan memberi informasi bahwa proses transaksi telah berhasil



Jika pengguna memasukkan PIN yang salah program akan meminta pengguna untuk memasukkan PIN yang benar.



Jika pengguna salah memasukkan PIN sampai 3 kali berturut - turut maka rekening tabungan pengguna akan di-*suspend* oleh program, dan mengirim pesan berisi bahwa proses *unsuspend* akan segera diproses oleh pengguna diba.



```

    } Reference
private void buttonConfirm_Click(object sender, EventArgs e)
{
    try
    {
        if (jumlahGagal < 3)
        {
            check = User.PinCheck(tabungan, textBoxPin.Text);
            if (check == true)
            {
                Transaction.TambahTransaksi(transaction);
                //Penambahan poin
                User currentUser = User.FetchDataByID(tabungan.User.Nik);
                currentUser.Points += 1;

                //upgrading user type
                if (currentUser.Points > currentUser.UserType.EndValue)
                {
                    currentUser.UserType = UserType.FetchDataByID(currentUser.UserType.IdType + 1);
                }
                User.UpdateData(currentUser);

                MessageBox.Show("transaksi berhasil");
                if (formTambahTransaksi.celkMerchant == true)
                {
                    Merchant.PromoMerc(transaction);
                }
                Inbox inbox = new Inbox(Inbox.GenerateID(), Inbox.GenerateInbox(transaction), transaction.TransactionDate, "Unread", transaction.TransactionDate, tabungan.User);
                inbox.AddData(inbox);
                this.Close();
            }
            else
            {
                jumlahGagal++;
                MessageBox.Show("pin salah ,mohon masukkan pin yang benar");
            }
        }
        else
        {
            tabungan.Status = "Suspended";
            Saving.UpdateData(tabungan);
            MessageBox.Show("Mohon maaf tabungan anda dalam status suspended " +
                           "Aplikasi akan segera mengirim permintaan ke pegawai dibutuhkan untuk mengaktifasi akun anda");
            Application.Exit();
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show("telah terjadi error pada sistem pesan kesalahan " + ex.Message);
    }
}

```

Berikut adalah *code* yang dijalankan ketika pengguna melakukan konfirmasi PIN.

I. FormRiwayatTransaksi

	IdTransaction	TransactionDate	Nominal	Description	Source
▶	11	14/01/2023 19:06	1000000	Menambah Loan	202261
	15	14/01/2023 22:52	1000000	Konfirm Deposito	202385
	16	14/01/2023 22:55	950000	Tutup Deposito	202385
	17	14/01/2023 22:56	1000000	Bayar Loan	202385
	18	14/01/2023 23:02	1000000	20227891	202385

Form ini akan menampilkan riwayat transaksi pengguna. Pengguna juga dapat memilih tipe transaksi apa yang ingin ditampilkan.

```
FormMain formMain;
List<TransactionType> listOfTypeTransaction = new List<TransactionType>();
List<Transaction> listOfTransactions = new List<Transaction>();

1 reference
private void FormRiwayatTransaksi_Load(object sender, EventArgs e)
{
    formMain = (FormMain)this.MdiParent;
    listOfTypeTransaction = TransactionType.ReadData();
    comboBoxJenisTransaksi.DataSource = listOfTypeTransaction;
    comboBoxJenisTransaksi.DisplayMember = "Name";

    listOfTransactions = Transaction.ReadData("", formMain.saving);
    if (listOfTransactions.Count > 0)
    {
        dataGridViewRiwayatTransaksi.DataSource = listOfTransactions;
    }
    else
    {
        dataGridViewRiwayatTransaksi.DataSource = null;
    }
}

1 reference
private void comboBoxJenisTransaksi_SelectedIndexChanged(object sender, EventArgs e)
{
    TransactionType selectedType = (TransactionType)comboBoxJenisTransaksi.SelectedItem;
    listOfTransactions = Transaction.ReadData($"{selectedType.IdTransType}", formMain.saving);
    if (listOfTransactions.Count > 0)
    {
        dataGridViewRiwayatTransaksi.DataSource = listOfTransactions;
    }
    else
    {
        dataGridViewRiwayatTransaksi.DataSource = null;
    }
}

1 reference
private void buttonKeluar_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Berikut adalah *code* dari FormRiwayatTransaksi

3.1.3 CRUD Tabungan Reguler

A. Class Savings

```
private string accountNumber;
private double balance;
private string status;
private string description;
private DateTime createDate;
private DateTime? updateDate;
private User user;
private Employee verifier;
private string type;
private List<Transaction> listTransactions;
```

Untuk memenuhi kebutuhan CRUD dari saving kami membuat 1 *class* baru bernama saving.

B. Read Saving

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- BacaData
- FetchDataById
- FetchDataByNik

BacaData

```
0 references
public static List<Saving> ReadData(string kriteria, string nilaiKriteria, string nilaiKriteriaKedua)// Read Data yang menerima 2 nilai kriteria
{
    string sql = "SELECT * FROM savings;";

    if (kriteria != "")
    {
        sql = $"SELECT * FROM savings WHERE {kriteria} LIKE '%{nilaiKriteria}%' OR {kriteria} LIKE '%{nilaiKriteriaKedua}%'";
    }

    MySqlDataReader hasil = Connection.ExecuteReader(sql);

    List<Saving> listSavings = new List<Saving>();

    while (hasil.Read() == true)
    {
        User user = User.FetchDataByID(hasil.GetValue(6).ToString());
        Employee verifier;
        DateTime? updateDate;
        if (hasil.GetValue(7).ToString() == "")
        {
            verifier = null;
        }
        else
        {
            verifier = Employee.FetchDataByID((int)hasil.GetValue(7));
        }
        if (hasil.GetValue(5).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(5);
        }
        Saving saving = new Saving(hasil.GetValue(0).ToString(), (double)hasil.GetValue(1), hasil.GetValue(2).ToString(),
            hasil.GetValue(3).ToString(), (DateTime)hasil.GetValue(4), updateDate, user, verifier, hasil.GetValue(8).ToString());
        listSavings.Add(saving);
    }
}

return listSavings;
}
```

Method BacaData berfungsi untuk membaca tabungan dari database

FetchDataByID

```
0 references
public static Saving FetchByID(string accountNumber)
{
    string sql = "select * from savings " +
        "where account_number = '" + accountNumber + "'";

    using (MySqlDataReader hasil = Connection.ExecuteQuery(sql))
    {
        if (hasil.Read() == true)
        {
            User user = User.FetchDataByID(hasil.GetValue(6).ToString());
            Employee verifier;
            DateTime? updateDate;
            if (hasil.GetValue(7).ToString() == "")
            {
                verifier = null;
            }
            else
            {
                verifier = Employee.FetchDataByID((int)hasil.GetValue(7));
            }
            if (hasil.GetValue(5).ToString() == "")
            {
                updateDate = null;
            }
            else
            {
                updateDate = (DateTime)hasil.GetValue(5);
            }
            Saving saving = new Saving(hasil.GetValue(0).ToString(), (double)hasil.GetValue(1), hasil.GetValue(2).ToString(),
                hasil.GetValue(3).ToString(), (DateTime)hasil.GetValue(4), updateDate, user, verifier, hasil.GetValue(8).ToString());
        }
        return saving;
    }
    else
    {
        return null;
    }
}
```

Method ini akan mengembalikan objek rekening berdasarkan nomor rekening yang dimasukkan ke dalam *argument method*.

FetchDataByNik

```
0 references
public static Saving FetchbyDataNIK(string nik)
{
    string sql = "select * from savings " +
        "where id_users = '" + nik + "'";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if (hasil.Read() == true)
    {
        User user = User.FetchDataByID(hasil.GetValue(6).ToString());
        Employee verifier;
        DateTime? updateDate;
        if (hasil.GetValue(7).ToString() == "")
        {
            verifier = null;
        }
        else
        {
            verifier = Employee.FetchDataByID((int)hasil.GetValue(7));
        }
        if (hasil.GetValue(5).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(5);
        }
        Saving saving = new Saving(hasil.GetValue(0).ToString(), (double)hasil.GetValue(1), hasil.GetValue(2).ToString(),
            hasil.GetValue(3).ToString(), (DateTime)hasil.GetValue(4), updateDate, user, verifier, hasil.GetValue(8).ToString());

        return saving;
    }
    else
    {
        return null;
    }
}
```

Method ini akan mengembalikan objek rekening berdasarkan nomor NIK pengguna yang dimasukkan ke dalam *argument method*.

C. Create Saving

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

```
1 reference
public static void AddData(Saving saving)
{
    string command = $"INSERT INTO savings(account_number, balance, status, keterangan, create_date, " +
        $"id_users,tipe) VALUES ('{saving.AccountNumber}', {saving.Balance}, '{saving.Status}', " +
        $"{saving.Description}', '{saving.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $"'{saving.User.Nik}', {saving.Tipe}');";
    Connection.ExecuteNonQuery(command);
}
```

Method AddData berfungsi untuk menambah data tabungan baru ke dalam database.

D. Update Saving

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData
- UpdateSavingSend & Receive

UpdateData

```
6 references
public static void UpdateData(Saving saving)
{
    string sql = $"UPDATE savings SET balance = {saving.Balance}, " +
        $"status = '{saving.Status}', keterangan = '{saving.Description}', " +
        $"create_date = '{saving.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $"update_date = '{saving.UpdateDate?.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $"id_users = '{saving.User.Nik}', verifikator = {saving.Verifier.IdEmployee} WHERE account_number = '{saving.AccountNumber}';";
    Connection.ExecuteNonQuery(sql);
}
```

Method update saving akan berguna untuk mengubah data tabungan yang tersimpan di dalam database.

UpdateSavingSend & Receive

```
4 references
private static void UpdateSavingSend(double nilaiTransfer, string tujuanTabungan, Connection koneksi)//method untuk mengurangi saldo dari tabungan asal
{
    string sql1 = "update savings " +
        "set balance = balance - " + nilaiTransfer + " " +
        "where account_number = '" + tujuanTabungan + "' ";
    Connection.ExecuteNonQuery(sql1, koneksi);
}

private static void UpdateSavingReceive(double nilaiTransfer, string tujuanTabungan, Connection koneksi)// method untuk menambah saldo di tabungan tujuan
{
    string sql2 = "update savings " +
        "set balance = balance + " + nilaiTransfer + " " +
        "where account_number = '" + tujuanTabungan + "' ";
    Connection.ExecuteNonQuery(sql2, koneksi);
}
```

Method ini berada di dalam *class* transaksi. Dimana *method* UpdateSavingSend bertujuan untuk mengurangi nominal saldo rekening asal dan UpdateSavingReceive bertujuan untuk menambah nominal saldo rekening tujuan

E. Delete Saving

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- DeleteData

DeleteData

```
0 references
public static bool DeleteData(Saving saving)
{
    string command = $"DELETE FROM saving WHERE account_number = {saving.AccountNumber};";
    int rowAffected = Connection.ExecuteNonQuery(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Delete saving berfungsi untuk menghapus tabungan yang dipilih dari database.

F. Generate Nomor Rekening

```
0 references
public static string GenerateID(User user)
{
    string command = $"SELECT YEAR(create_date), nik FROM users WHERE nik = '{user.Nik}';";
    string hasilID = "";
    MySqlDataReader hasil = Connection.ExecuteReader(command);

    if (hasil.Read() == true)
    {
        string newID = $"{hasil.GetValue(0).ToString()}{user.Nik.Substring(user.Nik.Length - 4)}";
        hasilID = newID;
    }
    return hasilID;
}
```

Fungsi GenerateID() berfungsi untuk menghasilkan nomor rekening berdasarkan tahun pembuatan rekening dan nomor NIK pemilik rekening. Format nomor rekening tabungan, yaitu “{tahun}{4 digit terakhir NIK}”.

G. FormTabunganReguler



FormTabunganReguler berfungsi untuk menampilkan informasi tentang tabungan reguler yang dimiliki oleh nasabah.

H. Kalkulasi Bunga Tabungan Reguler

```
// pengecekan bunga tabungan reguler dan pajak bunga (kalau ada)
bool cek = ModulDate(saving.CreateDate, 365);
if (cek == true)
{
    Saving.CalculateInterest(saving, double.Parse(settingsSection.Settings.Get("SavingInterest").Value.ValueXml.InnerText));
}

//proses biaya administrasi bulanan
bool cek2 = ModulDate(saving.CreateDate, 30);
if(cek2 == true)
{
    saving.Balance -= 10000;
    Saving.UpdateData(saving);
}
```

Kalkulasi pertumbuhan bunga tabungan reguler dilakukan dalam FormMain_Load. Sebelum menambahkan bunga, sistem akan mengecek apakah suatu tabungan sudah saatnya mendapat penambahan bunga dengan cara mengecek hari. Pengurangan biaya administrasi bulanan juga dilakukan dalam FormMain_Load. Setiap 30 hari, biaya admin akan dipotong dari tabungan reguler nasabah.

```
2 references
private bool ModulDate(DateTime savingCreated,int daysNumber)
{
    int days = (DateTime.Now - savingCreated).Days;
    if(days < daysNumber)
    {
        return false;
    }
    else
    {
        if(days%daysNumber != 0)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
}
```

Method ModulDate() yang memiliki 2 parameter berfungsi untuk mengecek apakah suatu tabungan sudah mencapai hari tertentu.

```
1 reference
public static void CalculateInterest(Saving saving,double value) // perhitungan bunga tabungan berdasarkan jumlah transaksi yang dilakukan di tahun saat ini
{
    string sql = "select count(source_account_number) as numberTransactions from transactions " +
                 "where source_account_number = '" + saving.AccountNumber + "' and " +
                 "year(transaction_date) = year(current_date) and id_transaction_type != 3 and id_transaction_type != 4 ";
    MySqlDataReader hasil = Connection.ExecuteReader(sql);

    if(hasil.Read()) == true
    {
        double interest = value * ((0.02 * saving.Balance) * double.Parse(hasil.GetValue(0).ToString()));
        if (interest > 50000)
        {
            Transaction transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now, interest, "Bunga", Saving.FetchByID("20226789"), saving, TransactionType.Transaction.TambahTransaksi(transaksi));

            double tax = 0.1 * interest;

            Transaction transaksi2 = new Transaction(Transaction.GenerateId(), DateTime.Now, tax, "Pajak Bunga", Saving.FetchByID("20226789"), saving, TransactionType.Transaction.TambahTransaksi(transaksi2));
        }
        else
        {
            Transaction transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now, interest, "Bunga", Saving.FetchByID("20226789"), saving, TransactionType.Transaction.TambahTransaksi(transaksi));
        }
    }
}
```

Method CalculateInterest() pada *class* Saving berfungsi untuk menghitung jumlah bunga yang didapat dari suatu tabungan reguler nasabah. *Method* ini akan dipanggil di FormMain_Load.

I. Konfirmasi Tabungan Reguler



Pegawai DiBa dapat mengonfirmasi tabungan reguler dari nasabah yang baru saja mendaftar menggunakan FormDaftarTabunganReguler.

```
private void FormDaftarTabunganReguler_Load(object sender, EventArgs e)
{
    formMain = (FormMain)this.MdiParent;
    listoSaving = Saving.ReadData("status", "Unverified", "Suspended", "Closing");
    if (listoSaving.Count > 0)
    {
        dataGridViewTabunganReguler.DataSource = listoSaving;
        if ((!dataGridViewTabunganReguler.Columns.Contains("btnConfirmGrid")) && (!dataGridViewTabunganReguler.Columns.Contains("btnUnsuspendGrid")) && (!dataGridViewTabunganReguler.Columns.Contains("btnClosingGrid")))
        {
            DataGridViewButtonColumn buttonColumn = new DataGridViewButtonColumn();
            buttonColumn.HeaderText = "Aksi";
            buttonColumn.Text = "Confirm";
            buttonColumn.Name = "btnConfirmGrid";
            buttonColumn.UseColumnTextForButtonValue = true;
            dataGridViewTabunganReguler.Columns.Add(buttonColumn);

            DataGridViewButtonColumn buttonColumn1 = new DataGridViewButtonColumn();
            buttonColumn1.HeaderText = "Aksi";
            buttonColumn1.Text = "Unsuspend";
            buttonColumn1.Name = "btnUnsuspendGrid";
            buttonColumn1.UseColumnTextForButtonValue = true;
            dataGridViewTabunganReguler.Columns.Add(buttonColumn1);

            DataGridViewButtonColumn buttonColumn2 = new DataGridViewButtonColumn();
            buttonColumn2.HeaderText = "Aksi";
            buttonColumn2.Text = "Closing";
            buttonColumn2.Name = "btnClosingGrid";
            buttonColumn2.UseColumnTextForButtonValue = true;
            dataGridViewTabunganReguler.Columns.Add(buttonColumn2);
        }
    }
}
```

Ketika FormDaftarTabunganReguler dimuat, maka seluruh data tabungan reguler yang memiliki status “Unverified”, “Suspended”, dan “Closing” akan dimuat dan dimasukkan ke data grid view. Dan setiap baris yang ada pada data grid view akan dibuatkan tombol “Confirm”, “Unsuspend”, dan “Closing” yang dapat digunakan oleh pegawai DiBa.

```
private void dataGridViewTabunganReguler_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == dataGridViewTabunganReguler.Columns["btnConfirmGrid"].Index && e.RowIndex >= 0)
    {
        try
        {
            Saving selectedSaving = (Saving)dataGridViewTabunganReguler.CurrentRow.DataBoundItem;
            if (selectedSaving.Status == "Unverified")
            {
                selectedSaving.Status = "Active";
                selectedSaving.Verificator = formMain.employee;
                selectedSaving.UpdateDate = DateTime.Now;
                Saving.UpdateData(selectedSaving);
                MessageBox.Show($"Sukses melakukan konfirmasi!");
                FormDaftarTabunganReguler_Load(buttonKeluar, e);
            }
            else
            {
                throw (new ArgumentException("Hanya tabungan dengan status 'Unverified' yang dapat dikonfirmasi!"));
            }
        }
        catch (Exception exception)
        {
            MessageBox.Show($"Gagal melakukan konfirmasi! Pesan kesalahan: {exception.Message}");
        }
    }
}
```

Ketika tombol “Confirm” ditekan, sistem akan mengubah status tabungan yang dipilih menjadi “Active”, mencatat data pegawai yang memverifikasi, dan mencatat tanggal perubahan. Sistem hanya melakukan perintah ini jika status tabungan masih “Unverified”.

```
        }  
    else if (e.ColumnIndex == dataGridViewTabunganReguler.Columns["btnUnsuspendGrid"].Index && e.RowIndex >= 0)  
{  
    try  
    {  
        Saving selectedSaving = (Saving)dataGridViewTabunganReguler.CurrentRow.DataBoundItem;  
        if (selectedSaving.Status == "Suspended")  
        {  
            selectedSaving.Status = "Active";  
            selectedSaving.Verifikator = formMain.employee;  
            selectedSaving.UpdateDate = DateTime.Now;  
            Saving.UpdateData(selectedSaving);  
            MessageBox.Show($"Sukses melakukan Unsuspend!");  
            FormDaftarTabunganReguler_Load(buttonKeluar, e);  
        }  
        else  
        {  
            throw (new ArgumentException("Hanya tabungan dengan status 'Suspended' yang dapat di-Unsuspend!"));  
        }  
    }  
    catch (Exception exception)  
    {  
        MessageBox.Show($"Gagal melakukan unsuspend! Pesan kesalahan: {exception.Message}");  
    }  
}
```

Ketika tombol “Unsuspend” ditekan, sistem akan mengubah status tabungan yang dipilih menjadi “Active”, mencatat data pegawai yang memverifikasi, dan mencatat tanggal perubahan. Sistem hanya melakukan perintah ini jika status tabungan masih “Suspended”.

```
        }  
    else if (e.ColumnIndex == dataGridViewTabunganReguler.Columns["btnClosingGrid"].Index && e.RowIndex >= 0)  
{  
    try  
    {  
        Saving selectedSaving = (Saving)dataGridViewTabunganReguler.CurrentRow.DataBoundItem;  
        if (selectedSaving.Status == "Closing")  
        {  
            selectedSaving.Status = "Closed";  
            Saving.UpdateData(selectedSaving);  
            MessageBox.Show($"Sukses melakukan Penutupan Tabungan Reguler");  
            FormDaftarTabunganReguler_Load(buttonKeluar, e);  
        }  
        else  
        {  
            throw (new ArgumentException("Hanya tabungan dengan status 'Closing' yang dapat ditutup !"));  
        }  
    }  
    catch (Exception exception)  
    {  
        MessageBox.Show($"Gagal melakukan unsuspend! Pesan kesalahan: {exception.Message}");  
    }  
}
```

Ketika tombol “Close” ditekan, sistem akan mengubah status tabungan yang dipilih menjadi “Closed”, mencatat data pegawai yang memverifikasi, dan mencatat tanggal perubahan. Sistem hanya melakukan perintah ini jika status tabungan masih “Closing”.

J. Penutupan Tabungan Reguler



Untuk menutup sebuah tabungan reguler, nasabah dapat menekan tombol “Tutup”. Ketika tabungan reguler telah ditutup, maka nasabah tidak dapat *login* kembali.

```
private void buttonTutup_Click(object sender, EventArgs e)
{
    DialogResult dialogResult = MessageBox.Show("Tabungan anda akan ditutup secara permanen, mohon pindahkan saldo yang anda miliki jika yak  
" "menutup tabungan", "Konfirmasi Penutupan Tabungan ", MessageBoxButtons.YesNo);

    if(dialogResult == DialogResult.Yes)
    {
        currentSaving.Status = "Closing";
        try
        {
            Saving.UpdateData(currentSaving);
            MessageBox.Show("Permintaan Penutupan Tabungan telah dikirim ke pegawai DiBa");
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Ketika nasabah menekan tombol “Tutup”, sistem akan memunculkan sebuah *message box* yang menanyakan tentang konfirmasi penutupan rekening. Setelah itu, sistem akan mengubah status tabungan reguler tersebut menjadi “Closing”. Setelah itu, proses konfirmasi penutupan akan diserahkan ke pegawai DiBa untuk dikonfirmasi penutupan.

3.1.4 CRUD Tabungan Deposito

A. Class Deposit

```
43 references
public class Deposit
{
    private string idDeposit;
    private string dueDate;
    private double nominal;
    private double interestRate;
    private string status;
    private DateTime createDate;
    private DateTime updateDate;
    private Saving saving;
    private Employee openVerifier;
    private Employee closeVerifier;
```

Kami membuat class deposit untuk memenuhi kebutuhan CRUD

B. Read Deposit

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- ReadData

ReadData

```
1 references
public static List<Saving> ReadData(string kriteria, string nilaiKriteria, string nilaiKriteriaMedua,string nilaiKriteriaMetiga)// Read Data yang menerima 3 nilai kriteria
{
    string sql = "SELECT * FROM savings;";
    if (kriteria != "")
    {
        sql = $"SELECT * FROM savings WHERE {kriteria} LIKE '%{nilaiKriteria}%' OR {kriteria} LIKE '%{nilaiKriteriaMedua}%' OR {kriteria} LIKE '%{nilaiKriteriaMetiga}%'";
    }

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    List<Saving> listSavings = new List<Saving>();
    while (hasil.Read() == true)
    {
        User user = User.FetchDataByID(hasil.GetValue(6).ToString());
        Employee verifier;
        DateTime? updateDate;
        if (hasil.GetValue(7).ToString() == "")
        {
            verifier = null;
        }
        else
        {
            verifier = Employee.FetchDataByID((int)hasil.GetValue(7));
        }
        if (hasil.GetValue(8).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(8);
        }
        Saving saving = new Saving(hasil.GetValue(0).ToString(), (double)hasil.GetValue(1), hasil.GetValue(2).ToString(),
        hasil.GetValue(3).ToString(), (DateTime)hasil.GetValue(4), updateDate, user, verifier, hasil.GetValue(5).ToString());
        listSavings.Add(saving);
    }
    return listSavings;
}

0 references
public static List<Saving> ReadData(string kriteria, string nilaiKriteria, string nilaiKriteriaMedua)// Read Data yang menerima 2 nilai kriteria
{
    string sql = "SELECT * FROM savings;";
    if (kriteria != "")
    {
        sql = $"SELECT * FROM savings WHERE {kriteria} LIKE '%{nilaiKriteria}%' OR {kriteria} LIKE '%{nilaiKriteriaMedua}%'";
    }

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    List<Saving> listSavings = new List<Saving>();
    while (hasil.Read() == true)
    {
        User user = User.FetchDataByID(hasil.GetValue(6).ToString());
        Employee verifier;
        DateTime? updateDate;
        if (hasil.GetValue(7).ToString() == "")
        {
            verifier = null;
        }
        else
        {
            verifier = Employee.FetchDataByID((int)hasil.GetValue(7));
        }
        if (hasil.GetValue(8).ToString() == "")
        {
            updateDate = null;
        }
        else
        {
            updateDate = (DateTime)hasil.GetValue(8);
        }
        Saving saving = new Saving(hasil.GetValue(0).ToString(), (double)hasil.GetValue(1), hasil.GetValue(2).ToString(),
        hasil.GetValue(3).ToString(), (DateTime)hasil.GetValue(4), updateDate, user, verifier, hasil.GetValue(5).ToString());
        listSavings.Add(saving);
    }
    return listSavings;
}
```

Method ReadData akan membaca data tabungan pengguna dari database. Method ini terdapat 2 varian, varian pertama menerima 3 kriteria sedangkan varian lainnya menerima 2 kriteria.

C. Create Deposit

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

```
1 references
public static void AddData(Deposit deposit)
{
    string command = $"INSERT INTO deposits(id_deposit, due_date, nominal, interest_rate, status, create_date, " +
    $" update_date, account_number, open_verifikator) VALUES ('{deposit.IdDeposit}', '{deposit.DueDate}', " +
    $" '{deposit.Nominal}', '{deposit.InterestRate}', '{deposit.Status}', '{deposit.CreateDate.ToString("yyyy-MM-dd")}', " +
    $" '{deposit.UpdateDate.ToString("yyyy-MM-dd")}', '{CultureInfo.GetCultureInfo("de-DE").Replace("#", "")}', " +
    $" '{deposit.Saving.AccountNumber}', '{deposit.OpenVerifikator.IdEmployee}')";
    Connection.ExecuteNonQuery(command);
}
```

Method add data berfungsi untuk menambah data baru deposit di dalam database

D. Update Deposit

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData
- CloseDeposit

UpdateData

```
0 references
public static void UpdateData(Deposit deposit)
{
    string command = $"UPDATE deposits SET due_date = '{deposit.DueDate}', nominal = {deposit.Nominal}, " +
        $" interest_rate = {deposit.InterestRate}, status = '{deposit.Status}', " +
        $" create_date = '{deposit.CreateDate.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $" update_date = '{deposit.UpdateDate?.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}', " +
        $" account_number = '{deposit.Saving.AccountNumber}', open_verifikator = '{deposit.OpenVerifier.IdEmployee}' " +
        $" WHERE id_deposit = '{deposit.IdDeposit}'";
    Connection.ExecuteDML(command);
}
```

Method UpdateData berfungsi untuk mengubah data tabungan deposit yang ada di dalam database.

CloseDeposit

```
1 reference
public static void CloseDeposit(Deposit deposit)
{
    string command = $"UPDATE deposits SET close_verifikator = {deposit.CloseVerifier.IdEmployee}, " +
        $"status = 'Closed', update_date = '{deposit.UpdateDate?.ToString("u", CultureInfo.GetCultureInfo("de-DE")).Replace("Z", "")}' " +
        $" WHERE id_deposit = '{deposit.IdDeposit}'";
    Connection.ExecuteDML(command);
}
```

Method CloseDeposit berfungsi untuk mengubah data tabungan deposit yang ada di dalam database.

E. Delete Deposit

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

- DeleteDeposit

DeleteDeposit

```
0 references
public static bool DeleteDeposit(Deposit deposit)
{
    string command = $"DELETE FROM deposits WHERE id_deposit = '{deposit.IdDeposit}';";
    int rowAffected = Connection.ExecuteDML(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Method DeleteDeposit berfungsi untuk mengubah data tabungan deposit yang ada di dalam database

F. FormDaftarDeposito



Form daftar deposito akan menampilkan daftar data deposito yang dimiliki oleh nasabah. Pada sisi pegawai, FormDaftarDeposito akan menampilkan daftar deposito dari nasabah yang memerlukan tindakan lebih lanjut, seperti konfirmasi pengajuan deposito dan konfirmasi penutupan deposito. Terdapat tiga tombol utama dalam form ini, yaitu tombol Tambah, Cairkan, dan Keluar. Tombol Tambah berfungsi untuk menambah pengajuan tabungan deposito dengan cara menampilkan FormTambahDeposito. Tombol Cairkan berfungsi untuk mengajukan pencairan deposito yang diinginkan dengan cara menampilkan FormPengajuanPencairan. Tombol Keluar berfungsi untuk menutup FormDaftarDeposito.

```
public bool IsInDateIntervals(Deposit deposit)
{
    DateTime dueDate = deposit.CreateDate;
    switch (deposit.DueDate)
    {
        case "1 Bulan":
            dueDate = dueDate.AddDays(30);
            break;
        case "3 Bulan":
            dueDate = dueDate.AddDays(90);
            break;
        case "6 Bulan":
            dueDate = dueDate.AddDays(180);
            break;
        case "1 Tahun":
            dueDate = dueDate.AddDays(365);
            break;
        case "2 Tahun":
            dueDate = dueDate.AddDays(730);
            break;
        case "3 Tahun":
            dueDate = dueDate.AddDays(1095);
            break;
    }
    return DateTime.Now >= deposit.CreateDate && DateTime.Now < dueDate;
}
```

Dalam FormDaftarDeposito terdapat *method* IsInDateIntervals() yang menerima parameter bertipe class Deposit. *Method* ini berfungsi untuk mengecek apakah sebuah deposito masih dalam rentang waktu jatuh temponya. Jika deposito masih dalam rentang waktu jatuh tempo, maka *method* ini akan mengembalikan nilai boolean *true*. Jika deposito tidak dalam rentang waktu jatuh tempo, maka *method* ini akan mengembalikan nilai boolean *false*.

```

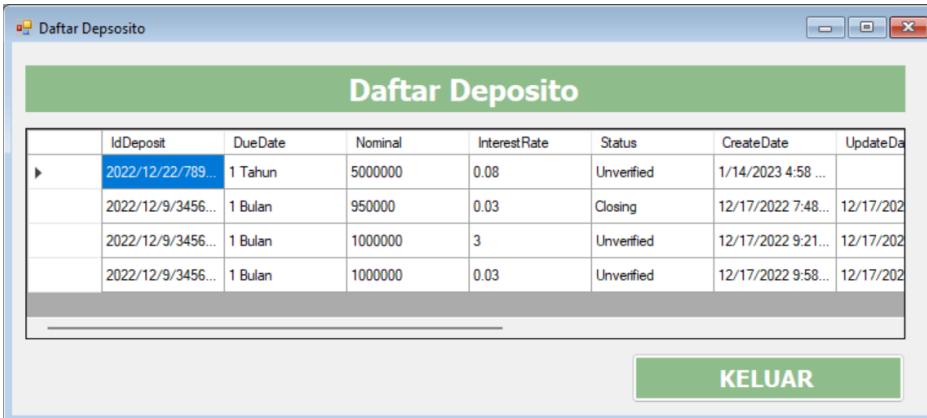
5 references
public void FormDaftarDeposito_Load(object sender, EventArgs e)
{
    formMain = (FormMain)this.MdiParent;
    saving = formMain.saving;
    if (formMain.user == null)
    {
        buttonTambah.Visible = false;
        buttonCair.Visible = false;
        listOfDeposits = Deposit.ReadData("status", "Unverified", "Closing");
        if (listOfDeposits.Count > 0)
        {
            dataGridViewInboxDeposito.DataSource = listOfDeposits;

            if (dataGridViewInboxDeposito.ColumnCount < 11)
            {
                DataGridViewButtonColumn buttonColumn = new DataGridViewButtonColumn();
                buttonColumn.HeaderText = "Aksi";
                buttonColumn.Text = "Confirm";
                buttonColumn.Name = "btnConfirmGrid";
                buttonColumn.UseColumnTextForButtonValue = true;
                dataGridViewInboxDeposito.Columns.Add(buttonColumn);

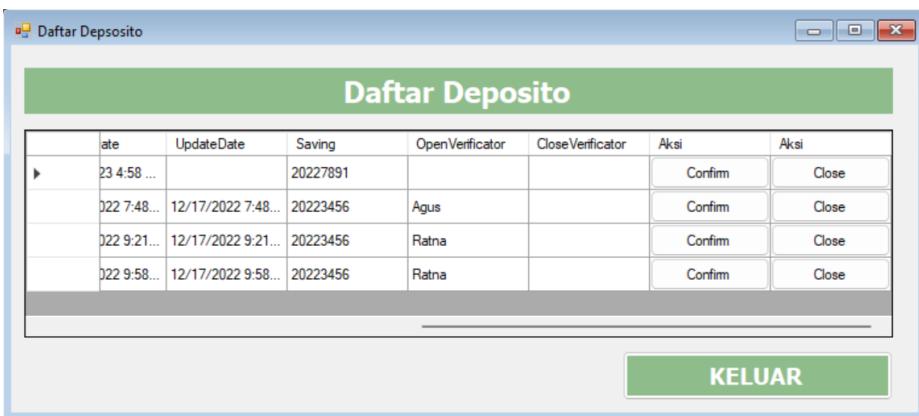
                DataGridViewButtonColumn buttonColumn1 = new DataGridViewButtonColumn();
                buttonColumn1.HeaderText = "Aksi";
                buttonColumn1.Text = "Close";
                buttonColumn1.Name = "btnCloseGrid";
                buttonColumn1.UseColumnTextForButtonValue = true;
                dataGridViewInboxDeposito.Columns.Add(buttonColumn1);
            }
        }
    }
    else
    {
        dataGridViewInboxDeposito.DataSource = null;
    }
}

```

Ketika FormDaftarDeposito dimuat, sistem akan mengecek *user* yang sedang *login*. Jika *user* tersebut adalah seorang pegawai DiBa, maka tombol “Tambah” dan tombol “Cairkan” akan dihilangkan karena kedua tombol tersebut hanya ditujukan untuk nasabah. Sedangkan, dua tombol baru akan ditambahkan pada setiap baris deposito. Kedua tombol ini adalah tombol “Confirm” dan tombol “Close”.

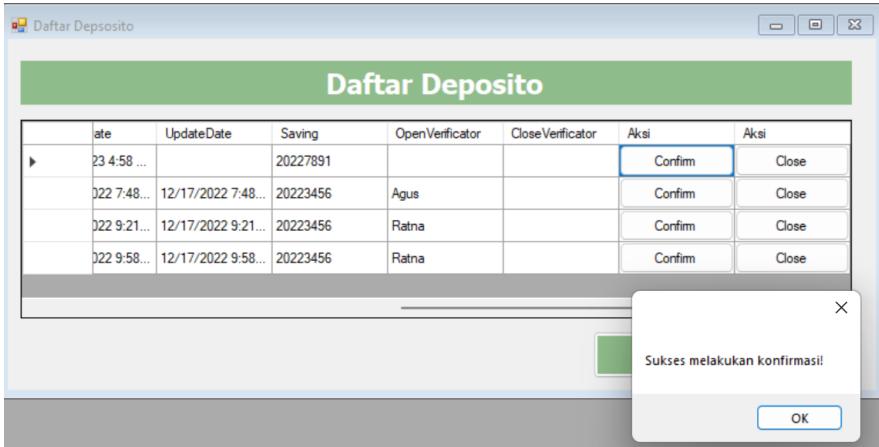


KELUAR



KELUAR

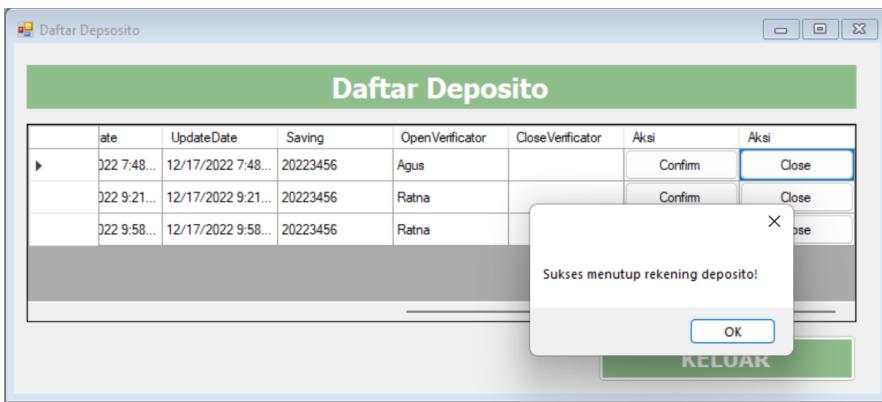
Gambar di atas merupakan tampilan FormDaftarDeposito dari sisi pegawai. Pegawai dapat melihat daftar deposito dari semua nasabah yang memerlukan tindakan lebih lanjut. Pegawai dapat menekan tombol Confirm untuk mengonfirmasi deposito nasabah yang berstatus Unverified dan pegawai dapat menutup deposito nasabah yang berstatus Closing dengan cara menekan tombol Close.



Ketika pegawai menekan tombol Confirm, maka Message Box seperti di atas akan muncul.

```
if (e.ColumnIndex == dataGridViewInboxDeposito.Columns["btnConfirmGrid"].Index && e.RowIndex >= 0)
{
    try
    {
        Deposit selectedDeposit = (Deposit)dataGridViewInboxDeposito.CurrentRow.DataBoundItem;
        if (selectedDeposit.Status == "Unverified")
        {
            selectedDeposit.Status = "Active";
            selectedDeposit.OpenVerifier = formMain.employee;
            selectedDeposit.UpdateDate = DateTime.Now;
            Deposit.UpdateData(selectedDeposit);
            merc = new Merchant();
            Transaction transaksi = new Transaction(Transaction.GenerateId(), (DateTime)selectedDeposit.UpdateDate, selectedDeposit.Nominal, "Konfirm Deposito", selectedDeposit.Saving, TransactionType.FetchDataByID(2), merc);
            Transaction.TambahTransaksi(transaksi, selectedDeposit, "konfirmDeposit");
            MessageBox.Show($"Sukses melakukan konfirmasi!");
            FormDaftarDeposito_Load(buttonKeluar, e);
        }
        else
        {
            throw (new ArgumentException("Nasabah tidak mengajukan pembukaan deposito!"));
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show($"Gagal melakukan konfirmasi! Pesan kesalahan: {exception.Message}");
    }
}
```

Ketika tombol “Confirm” ditekan, maka kode di atas akan dijalankan. Sistem akan membuat objek deposito yang dipilih berdasarkan baris pada data grid view. Sistem hanya akan mengonfirmasi tabungan deposito jika status tabungan tersebut adalah “Unverified”. Sistem akan mengubah status deposito yang dipilih menjadi “Active” dan menambah data pegawai yang melakukan verifikasi ke dalam data OpenVerifier deposito. Tanggal perubahan juga dicatat. Setelah itu, proses transaksi dilakukan. Sistem akan mentransfer dana tabungan reguler nasabah ke tabungan deposito nasabah sebesar nilai nominal deposito yang diajukan.



Message Box seperti di atas akan muncul jika pegawai menutup sebuah tabungan deposito.

```

else if (e.ColumnIndex == dataGridViewInboxDeposito.Columns["btnCloseGrid"].Index && e.RowIndex >= 0)
{
    try
    {
        Deposit selectedDeposit = (Deposit)dataGridViewInboxDeposito.CurrentRow.DataBoundItem;
        if (selectedDeposit.Status == "Closing")
        {
            selectedDeposit.CloseVerifier = formMain.employee;
            selectedDeposit.UpdateDate = DateTime.Now;
            Deposit.UpdateData(selectedDeposit);
            Deposit.CloseDeposit(selectedDeposit);
            merc = new Merchant();
            Transaction transaksi = new Transaction(Transaction.GenerateId(), (DateTime)selectedDeposit.UpdateDate, selectedDeposit.Nominal, "Tutup Deposito", selectedDeposit.Saving, TransactionType.FetchDataByID(2), merc);
            Transaction.TambahTransaksi(transaksi, selectedDeposit, "tutupDeposito");
            MessageBox.Show($"Sukses menutup rekening deposito!");
            FormDaftarDeposito_Load(buttonKeluar, e);
        }
        else
        {
            throw (new ArgumentException("Nasabah tidak mengajukan pencairan!"));
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show($"Gagal menutup rekening deposito! Pesan kesalahan: {exception.Message}");
    }
}

```

Ketika pegawai menekan tombol “Close”, maka kode di atas akan dijalankan. Sistem akan mengambil data deposito berdasarkan baris dari data grid view. Sistem hanya akan menutup tabungan deposito dengan status “Closing”. Setelah itu, sistem akan mencatat data pegawai DiBa yang menutup tabungan deposito tersebut dan mencatat tanggal perubahannya. Setelah itu, sistem akan memindahkan saldo tabungan deposito tersebut ke tabungan reguler nasabah sebesar nilai deposito yang ada.



Gambar di atas adalah tampilan dari FormDaftarDeposito dari sisi nasabah.

```

        }
        else
        {
            buttonTambah.Visible = true;
            buttonCair.Visible = true;
            listOfDeposits = Deposit.ReadData("account_number", $"{formMain.saving.AccountNumber}", $"{ formMain.saving.AccountNumber}");

            if (listOfDeposits.Count > 0)
            {
                dataGridViewInboxDeposito.DataSource = listOfDeposits;

                foreach (Deposit deposit in listOfDeposits)
                {
                    if (IsInDateIntervals(deposit) == false)
                    {
                        MessageBox.Show($"Deposit {deposit.IdDeposit} - {deposit.DueDate} siap untuk dicairkan!");
                    }
                }
            }
            else
            {
                dataGridViewInboxDeposito.DataSource = null;
            }
        }
    }
}

```

Gambar di atas merupakan potongan kode dari *method* FormDepositoLoad(). Kode tersebut akan berjalan apabila *user* yang *login* adalah seorang nasabah.



Jika seorang nasabah memiliki deposito yang siap dicairkan, maka akan muncul sebuah Message Box yang menginformasikan data deposito yang siap untuk dicairkan seperti gambar di atas ini.

G. FormTambahDeposito





FormTambahDeposito berfungsi untuk memasukkan data deposito yang ingin dibuat. Nasabah harus memasukkan nilai nominal deposito dan waktu tenggat atau jatuh tempo dari deposito tersebut. Setelah mengisi data yang diperlukan, nasabah dapat menekan tombol Tambah untuk mengirim pengajuan deposito. Setelah itu, nasabah harus menunggu hingga deposito tersebut telah disetujui oleh pegawai DiBa.

```

private void buttonTambah_Click(object sender, EventArgs e)
{
    try
    {
        formDaftarDeposito = (FormDaftarDeposito)this.Owner;
        double nominal = double.Parse(textBoxNominal.Text);
        string dueDate = (string)comboBoxWaktuTenggat.SelectedItem;

        //Saving saving = Saving.FetchByDataNIK(formMain.user.Nik);

        Configuration myConf = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
        ConfigurationSectionGroup userSettings = myConf.SectionGroups["userSettings"];
        var settingsSection = userSettings.Sections["DiBa_App.savingInformation"] as ClientSettingsSection;
        double interestRate = 0;
        switch (dueDate)
        {
            case "1 Bulan":
                interestRate += double.Parse(settingsSection.Settings.Get("SatuBulan").Value.ValueXml.InnerText);
                break;
            case "3 Bulan":
                interestRate += double.Parse(settingsSection.Settings.Get("TigaBulan").Value.ValueXml.InnerText);
                break;
            case "6 Bulan":
                interestRate += double.Parse(settingsSection.Settings.Get("EnamBulan").Value.ValueXml.InnerText);
                break;
            case "1 Tahun":
                interestRate += double.Parse(settingsSection.Settings.Get("SatuTahun").Value.ValueXml.InnerText);
                break;
            case "2 Tahun":
                interestRate += double.Parse(settingsSection.Settings.Get("DuaTahun").Value.ValueXml.InnerText);
                break;
            case "3 Tahun":
                interestRate += double.Parse(settingsSection.Settings.Get("TigaTahun").Value.ValueXml.InnerText);
                break;
        }

        if (formDaftarDeposito.saving.Status != "Unverified")
        {
            if (formDaftarDeposito.saving.Balance - nominal >= 50000)
            {
                Deposit newDeposit = new Deposit(Deposit.GenerateID(formDaftarDeposito.saving), dueDate, nominal, interestRate, "Unverified", DateTime.Now, null, formDaftarDeposito.saving, null, null);
                Deposit.AddData(newDeposit);
                MessageBox.Show("Pengajuan deposito berhasil dilakukan!");
            }
            else
            {
                throw (new ArgumentException("Saldo tidak cukup!"));
            }
        }
        else
        {
            throw (new ArgumentException("Status tabungan masih 'Unverified'!"));
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show($"Gagal mengajukan deposito! Pesan kesalahan: {exception.Message}");
    }
}

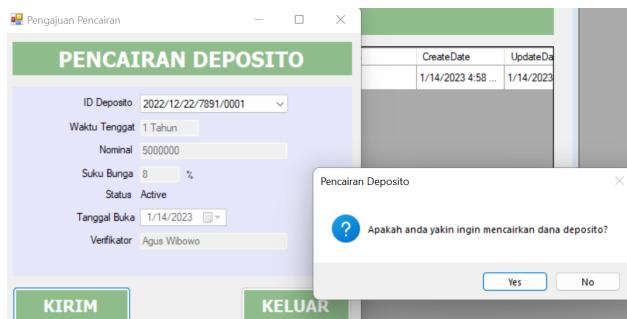
```

Ketika tombol Tambah ditekan, maka kode di atas akan dijalankan. Sistem akan mengambil nilai nominal deposito dan waktu jatuh tempo. Sistem juga mengambil nilai suku bunga deposito dari savingInformation.settings berdasarkan waktu jatuh tempo. Setelah itu, sistem akan mengecek apakah status tabungan reguler nasabah. Jika statusnya adalah Active dan jumlah nominal saldo tabungan reguler nasabah melebihi nilai nominal deposito, maka objek Deposito akan dibuat dan ditambahkan. Jika terjadi kesalahan, maka sebuah message box akan muncul yang berisi pesan kesalahan.

H. FormPengajuanPencairan



FormPengajuanPencairan berfungsi untuk memfasilitasi nasabah yang ingin mencairkan tabungan depositonya. Nasabah dapat memilih tabungan deposito yang ingin dicairkan dengan memilih ID Deposito pada sebuah combo box. Setelah itu, informasi tentang deposito yang dipilih akan muncul. Namun, nasabah hanya boleh mencairkan tabungan deposito yang berstatus Active. Selain dari status tersebut, tombol Kirim tidak akan dapat ditekan.



Ketika nasabah menekan tombol Kirim, sebuah message box akan keluar untuk mengonfirmasi pilihan pencairan nasabah. Jika nasabah ingin mencairkan tabungan deposito sebelum masa jatuh temponya, maka sebuah message box yang menginformasikan tentang adanya biaya *penalty* akan muncul.

```
1 reference
private void FormPengajuanPencairan_Load(object sender, EventArgs e)
{
    formDaftarDeposito = (FormDaftarDeposito)this.Owner;
    comboBoxID.DataSource = formDaftarDeposito.listOfDeposits;
    comboBoxID.DisplayMember = "IdDeposit";
}
```

Ketika FormPengajuanPencairan dimuat, combo box ID akan mengambil dan menampilkan ID daftar deposit yang dimiliki oleh nasabah.

```

private void comboBoxID_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        Deposit selectedDeposit = (Deposit)comboBoxID.SelectedItem;

        if (selectedDeposit.Status == "Active")
        {
            buttonCair.Enabled = true;
        }
        else
        {
            buttonCair.Enabled = false;
        }
        textBoxDueDate.Text = selectedDeposit.DueDate;
        textBoxNominal.Text = selectedDeposit.Nominal.ToString();
        textBoxInterestRate.Text = $"{selectedDeposit.InterestRate * 100}";
        labelStatus.Text = selectedDeposit.Status;
        dateTimePickerCreateDate.Value = selectedDeposit.CreateDate;
        if(selectedDeposit.OpenVerifier != null)
        {
            textBoxVerifier.Text = $"{selectedDeposit.OpenVerifier.FirstName} {selectedDeposit.OpenVerifier.LastName}";
        }
        else
        {
            textBoxVerifier.Text = "-";
        }
    }
    catch(Exception exception)
    {
        MessageBox.Show($"Gagal melakukan pengajuan pencairan! Pesan kesalahan: {exception.Message}");
    }
}

```

Ketika terjadi perubahan pada combo box ID, maka sistem akan menampilkan informasi deposit sesuai ID yang dipilih pada combo box.

```

private void buttonCair_Click(object sender, EventArgs e)
{
    try
    {
        Deposit selectedDeposit = (Deposit)comboBoxID.SelectedItem;
        if (selectedDeposit != null)
        {
            double penaltyAmount = 0;

            //Configuration
            Configuration myConf = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
            ConfigurationSectionGroup userSettings = myConf.SectionGroups["userSettings"];
            var settingsSection = userSettings.Sections["DiBa_App.savingInformation"] as ClientSettingsSection;

            DialogResult konfirmasi = MessageBox.Show(this, "Apakah anda yakin ingin mencairkan dana deposito?", "Pencairan Deposito", MessageBoxButtons.YesNo);
            if (konfirmasi == DialogResult.Yes)
            {
                if (formDaftarDeposito.IsInDateIntervals(selectedDeposit))
                {
                    DialogResult konfirmasiPenalty = MessageBox.Show(this, "Apakah anda yakin ingin mencairkan dana deposito meskipun belum mencapai waktu?", "Pencairan Deposito", MessageBoxButtons.YesNo);
                    if (konfirmasiPenalty == DialogResult.Yes)
                    {
                        penaltyAmount += selectedDeposit.Nominal * double.Parse(settingsSection.Settings.Get("Penalty").Value.ValueXml.InnerText);
                    }
                    else
                    {
                        return;
                    }
                }
                selectedDeposit.Status = "Closing";
                selectedDeposit.Nominal -= penaltyAmount;
                Deposit.UpdateData(selectedDeposit);
                MessageBox.Show("Pencairan deposito berhasil diajukan!");
            }
        }
        else
        {
            throw (new ArgumentException("Tidak ada rekening yang dapat dicairkan!"));
        }
    }
}

```

Ketika tombol Kirim ditekan, maka sistem akan mengambil data tabungan deposito berdasarkan data deposit yang ada pada combo box ID. Setelah itu, objek deposito tersebut akan dicek waktu tenggatnya dengan *method* IsInDateIntervals(). Jika deposito tersebut masih berada dalam rentang waktu jatuh tempo, maka sistem akan mengambil nilai *penalty* dari savingInformation.settings dan disimpan ke dalam variabel penaltyAmount. Namun, jika deposito tidak berada dalam rentang waktu jatuh tempo, maka nilai penaltyAmount sama dengan 0. Setelah itu, status deposit yang terpilih akan dirubah menjadi “Closing” dan nilai nominal deposit tersebut akan dikurangi dengan penaltyAmount.

I. Kalkulasi Deposito

```
Configuration myConf = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
ConfigurationSectionGroup userSettings = myConf.SectionGroups["userSettings"];
var settingsSection = userSettings.Sections["diba_App.savingInformation"] as ClientSettingsSection;

listOfDeposit = Deposit.ReadData("account_number", $"{saving.AccountNumber}", $"{saving.AccountNumber}");
foreach (Deposit deposit in listOfDeposit)
{
    if (formDaftarDeposito.IsInDateIntervals(deposit) && deposit.Status == "Active")
    {
        double taxRate = double.Parse(settingsSection.Settings.Get("Tax").Value.ValueXml.InnerText);
        Deposit.CalculateInterest(deposit, taxRate);
        Deposit.UpdateData(deposit);
    }
}
```

Kalkulasi deposito dimulai ketika FormMain dimuat. Sistem akan mengambil daftar deposit yang dimiliki oleh nasabah. Setelah itu, sistem akan mulai melakukan kalkulasi pertambahan bunga yang diperoleh oleh setiap deposit yang masih berada dalam rentang jatuh tempo.

```
1 reference
public static void CalculateInterest(Deposit deposit, double tax)
{
    if (deposit.Status == "Active")
    {
        double day = 0;
        double month = 0;
        int diffOfDays = (DateTime.Now - deposit.CreateDate).Days;
        int diffOfMonths = diffOfDays / 30;
        switch (deposit.DueDate)
        {
            case "1 Bulan":
                day += 30;
                month += 1;
                break;
            case "3 Bulan":
                day += 90;
                month += 3;
                break;
            case "6 Bulan":
                day += 180;
                month += 6;
                break;
            case "1 Tahun":
                day += 365;
                month += 12;
                break;
            case "2 Tahun":
                day += 730;
                month += 24;
                break;
            case "3 Tahun":
                day += 1095;
                month += 36;
                break;
        }
        for (int months = 0; months < diffOfMonths; months++)
        {
            double interestAmount = (deposit.Nominal * deposit.InterestRate * month) / 12;
            double taxAmount = interestAmount * tax;
            deposit.Nominal += interestAmount - taxAmount;
        }
    }
}
```

Kode di atas merupakan *method* CalculateInterest() yang dimiliki oleh *class* Deposit yang berfungsi untuk menghitung nilai pertumbuhan bunga pada deposito. Total bunga yang diraih dipengaruhi oleh jumlah nominal deposito yang ada dan nilai pajak.

4 Milestone Tahap 4

4.1 Merchant

A. Class Merchant

Entitas merchants berfungsi untuk menyimpan data promo yang digunakan ketika bertransaksi dengan merchant . Promo ini dapat dipakai oleh user ketika bertransaksi dengan merchant. Untuk memenuhi kebutuhan CRUD kami membuat 1 *class* baru bernama merchant. Tiap kali user memakai promo merchant maka user akan mendapatkan tambahan saldo sebesar 10% dari nominal transaksi.

```
private int id;
private string name;
private string promoCode;
private DateTime startDate;
private DateTime endDate;
```

Berikut adalah data member dari *class* merchant.

B. Create Merchant

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

Hanya karyawan DiBa saja yang memiliki akses untuk menambah merchant. Proses menambah merchant akan menggunakan *method* AddData. *Method* ini akan dipanggil di form tambah merchant. Form tambah merchant hanya bisa dipanggil oleh karyawan DiBa.

```
public static void AddData(Merchant merc)
{
    string sql = "insert into merchants(id,name,promo_code,start_date,end_date) " +
    "values(" + merc.Id + ", " + merc.Name + ", " + merc.PromoCode + ", " + merc.StartDate.ToString("yyyy-MM-dd H:mm:ss") + ", " +
    merc.EndDate.ToString("yyyy-MM-dd H:mm:ss") + ")";
    Connection.ExecuteNonQuery(sql);
}
```

C. Read Merchant

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- BacaData
- FetchDataByID
- CariPromo

BacaData

```
1 references
public static List<Merchant> BacaData(string kriteria , string nilaiKriteria)
{
    string sql;
    if(kriteria == "")
    {
        sql = "select * from merchants";
    }
    else
    {
        sql = "select * from merchants "
        "where " + kriteria + " like '%" + nilaiKriteria + "%'";
    }

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    List<Merchant> listMerchant = new List<Merchant>();

    while(hasil.Read() == true)
    {
        Merchant m = new Merchant(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(), hasil.GetValue(2).ToString(),
                                    DateTime.Parse(hasil.GetValue(3).ToString()), DateTime.Parse(hasil.GetValue(4).ToString()));

        listMerchant.Add(m);
    }
}

return listMerchant;
```

Proses membaca data promo merchant dari database akan menggunakan *method* BacaData. *Method* ini akan dipanggil ketika form daftar merchant dipanggil. Form daftar merchant hanya bisa dipanggil oleh karyawan DiBa.

FetchDataByID

```
2 references
public static Merchant FetchDataByID(int id)
{
    string sql = "select * from merchants "
    "where id = " + id;
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if(hasil.Read()==true)
    {
        Merchant merchant = new Merchant(int.Parse(hasil.GetValue(0).ToString()) ,hasil.GetValue(1).ToString(),
                                         hasil.GetValue(2).ToString() , DateTime.Parse(hasil.GetValue(3).ToString()),
                                         DateTime.Parse(hasil.GetValue(4).ToString()));

        return merchant;
    }
    else
    {
        Merchant merchant = new Merchant();
        return merchant;
    }
}
```

Method ini akan membaca data promo merchant berdasarkan id.

CariPromo

```
2 references
public static Merchant CariPromo(string kodePromo)
{
    string sql = "select * from merchants "
    "where promo_code =" + kodePromo + "";

    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if (hasil.Read() == true)
    {
        Merchant merchant = new Merchant(int.Parse(hasil.GetValue(0).ToString()), hasil.GetValue(1).ToString(),
                                         hasil.GetValue(2).ToString() , DateTime.Parse(hasil.GetValue(3).ToString()),
                                         DateTime.Parse(hasil.GetValue(4).ToString()));

        return merchant;
    }
    else
    {
        return null;
    }
}
```

Method ini akan membaca data promo merchant berdasarkan kode promo.

D. Update Merchant

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData

UpdateData

```
1 reference
public static void UpdateData(Merchant merc)
{
    string sql = "update merchants " +
        "set promo_code = '" + merc.PromoCode + "', " +
        "start_date = '" + merc.StartDate.ToString("yyyy-MM-dd H:mm:ss") + "','" +
        "end_date = '" + merc.EndDate.ToString("yyyy-MM-dd H:mm:ss") + "' " +
        " where id = " + merc.Id;
    Connection.ExecuteDML(sql);
}
```

Proses mengubah data promo merchant akan menggunakan *method* update data. *Method* ini akan dipanggil di form ubah merchant. Form ini akan dipanggil ketika button ubah di klik. Button ubah berada di dalam form ubah merchant.

E. Daftar Promo



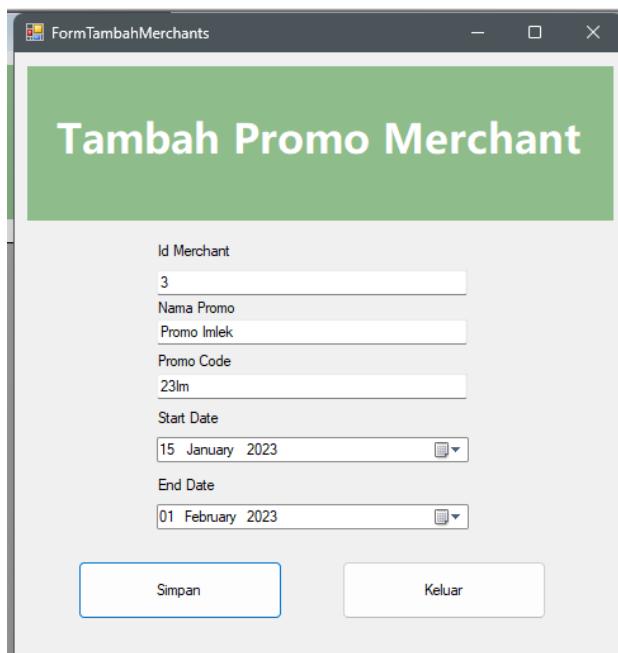
Baik pengguna maupun pegawai dapat mengakses daftar promo merchant yang ada di dalam database.

F. GenerateId

```
1 reference
public static int GenerateId()
{
    string sql = "select max(id) from merchants ";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);
    int hasilKode;
    if (hasil.Read() == true && hasil.GetValue(0).ToString() != "")
    {
        hasilKode = int.Parse(hasil.GetValue(0).ToString()) + 1;
    }
    else
    {
        hasilKode = 1;
    }
    return hasilKode;
}
```

Method ini digunakan untuk menghasilkan kode id terbaru untuk promo merchant.

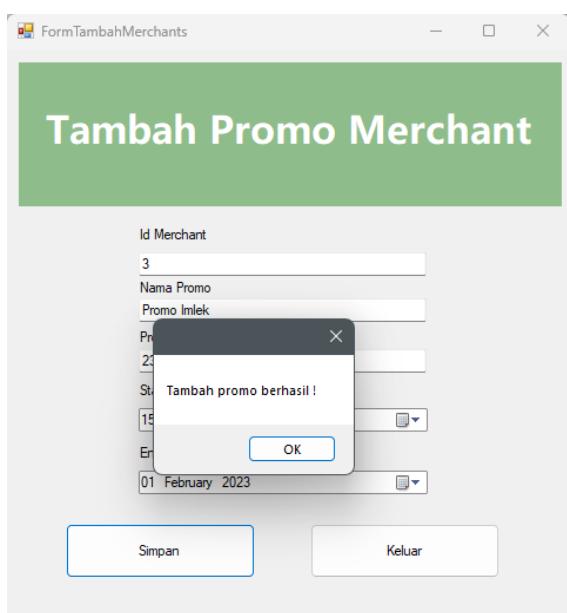
G. Tambah Promo



Hanya pegawai diba yang memiliki akses untuk menambah promo *merchant*, jika kode promo melebih 6 karakter program akan mengeluarkan pesan exception.



Program akan memberikan pesan penambahan promo berhasil jika panjang kode promo tidak lebih dari 6 karakter.



```

    I reference
private void FormTambahMerchants_Load(object sender, EventArgs e)
{
    textBoxIdMerchant.Text = Merchant.GenerateId().ToString();
}

I reference
private void buttonSimpan_Click(object sender, EventArgs e)
{
    try
    {
        Merchant merchant = new Merchant(int.Parse(textBoxIdMerchant.Text),
                                         textBoxNamaMerchant.Text,
                                         textBoxPromoCode.Text, dateTimePickerStartDate.Value, dateTimePickerEndDate.Value);

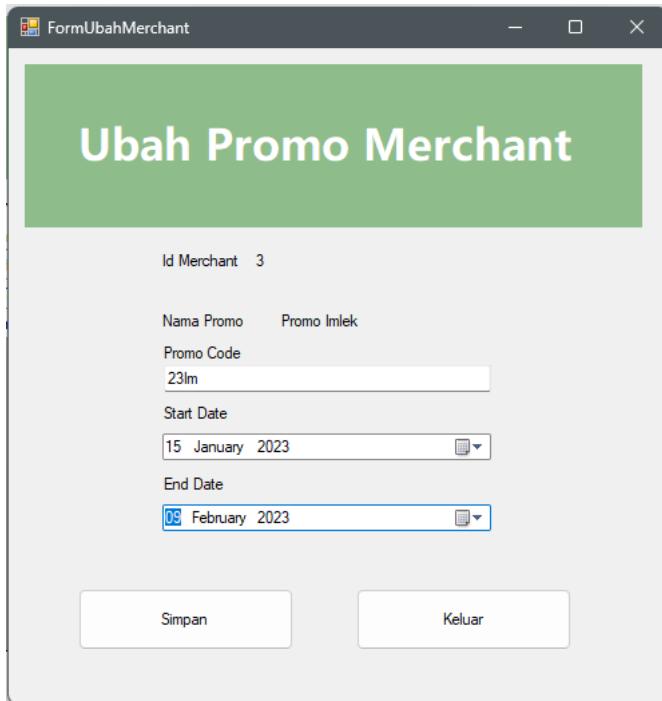
        Merchant.AddData(merchant);
        MessageBox.Show("Tambah promo berhasil !");
        this.Close();
    }
    catch(Exception ex)
    {
        MessageBox.Show("pesan kesalahan : " + ex.Message);
    }
}

I reference
private void buttonKeluar_Click(object sender, EventArgs e)
{
    this.Close();
}

```

Berikut adalah *code* yang dijalankan ketika pegawai menambah promo baru.

H. Ubah Promo



Hanya pegawai yang bisa mengubah informasi promo di dalam database. Informasi yang dapat diubah adalah kode promo, tanggal mulai dan tanggal berakhirnya promo tersebut.

```

private void FormubahMerchant_Load(object sender, EventArgs e)
{
    formDataForMerc = (FormDaftarMerchants)this.Owner;
    string formDaftarMerc.merc;
    labelIdMerchant.Text = formDaftarMerc.Id.ToString();
    labelNama.Text = merc.Name;
    textBoxPromoCode.Text = merc.PromoCode;
    dateTimePickerStartDate.Value = merc.StartDate;
    dateTimePickerEndDate.Value = merc.EndDate;
}

reference
private void buttonSimpan_Click(object sender, EventArgs e)
{
    try
    {
        Merchant merc = new Merchant(int.Parse(labelIdMerchant.Text), labelNama.Text, textBoxPromoCode.Text, dateTimePickerStartDate.Value,
            dateTimePickerEndDate.Value);
        Merchant.UpdateData(merc);
        MessageBox.Show("update data berhasil !");
    }
    catch (Exception ex)
    {
        MessageBox.Show("pesan kesalahan : " + ex.Message);
    }
}

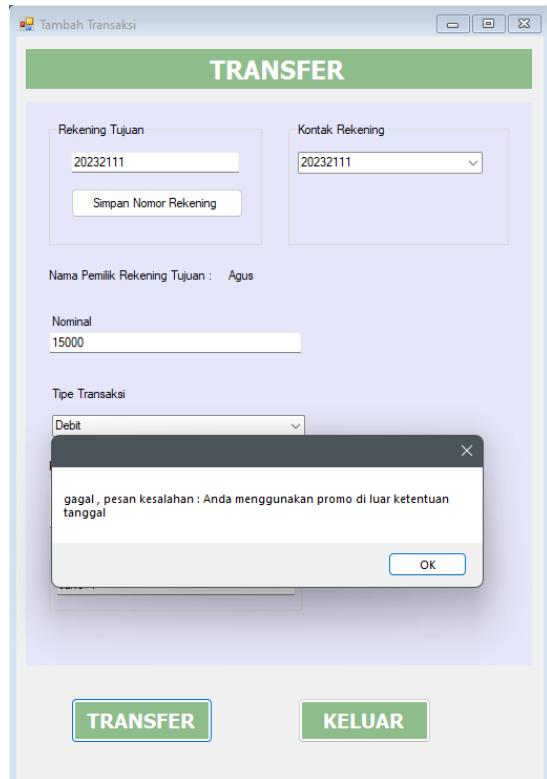
reference
private void buttonKeluar_Click(object sender, EventArgs e)
{
    this.Close();
}

```

Berikut adalah *code* yang dijalankan ketika pegawai mengubah data promo.

I. Penggunaan Promo

Promo merchant hanya bisa dipakai oleh pengguna jika rekening tujuan adalah rekening dengan tipe merchant.



Jika pengguna menggunakan promo yang sudah *expired*, maka program akan mengeluarkan pesan *exception*.

```

1 reference
public static void PromoMerc(Transaction transaksi )
{
    string sql = "update savings " +
        "set balance = balance + " + (0.1 * transaksi.Nominal) + " " +
        "where account_number = '" + transaksi.SourceAccount + "'";
    Connection.ExecuteNonQuery(sql);
}

```

Berikut adalah *method* PromoMerc. *Method* ini berfungsi untuk menambah saldo pengguna jika pengguna menggunakan promo merchant. Penambahan saldo tabungan adalah 10% dari nominal transaksi.

```

1 reference
public static bool CekTanggal(Merchant merc , DateTime now)
{
    if(now >= merc.StartDate && now <= merc.EndDate)
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

Berikut ini adalah *method* CekTanggal, *method* ini digunakan untuk melakukan pengecekan promo yang digunakan. Apakah tanggal pada saat promo dipakai masih berada dalam ketentuan tanggal promo atau tidak.

```

if (groupBoxPromoMerchant.Enabled == true)
{
    merc = Merchant.CariPromo(textBoxKodePromo.Text);
    bool cek = Merchant.CekTanggal(merc, DateTime.Now);
    if(cek == true)
    {
        cekMerchant = true;
    }
    else
    {
        merc = new Merchant();
        cekMerchant = false;
        throw (new ArgumentException("Anda menggunakan promo di luar ketentuan tanggal"));
    }
}
else
{
    merc = new Merchant();
    cekMerchant = false;
}

```

Method CekTanggal akan dipanggil ketika pengguna melakukan proses transaksi di FormTambahTransaksi.

4.2 Loan

A. Class Loan

Entitas loans di dalam erd berguna untuk mencatat hutang yang dibuat oleh pengguna. Relasi hubungan loans adalah one to one dengan saving. Ini berarti tiap 1 akun tabungan DiBa hanya bisa mempunyai 1 hutang di satu waktu. Sedangkan employees mempunyai relasi one to many dengan loans. Ini berarti 1 karyawan dapat mengkonfirmasi banyak pengajuan hutang. Untuk memenuhi kebutuhan loans , kami membuat class loan di dalam project DiBa_lib. Batas waktu pembayaran hutang adalah 6 bulan dari waktu pengajuan hutang. Dana pengajuan hutang akan diambil dari rekening khusus DiBa dengan keterangan “Diba Account” yang ada di dalam database. Ketika

pengguna melakukan pembayaran, nominal hutang yang dibayar juga akan masuk ke rekening khusus DiBa

20226789	9998897120	Active	DiBa Account	2022-12-22 15:36:55	2022-12-22 15:36:55	123456789	2	basic
----------	------------	--------	--------------	---------------------	---------------------	-----------	---	-------

```
3 references
public class Loan
{
    private int idLoan;
    private Saving lender;
    private Employee verifikator;
    private double amount;

    3 references
    public Loan(int idLoan, Saving lender, Employee verifikator, double amount)
    {
        IdLoan = idLoan;
        Lender = lender;
        Verifikator = verifikator;
        Amount = amount;
    }

    4 references
    public int IdLoan { get => idLoan; set => idLoan = value; }
    7 references
    public Saving Lender { get => lender; set => lender = value; }
    5 references
    public Employee Verifikator { get => verifikator; set => verifikator = value; }
    11 references
    public double Amount { get => amount; set => amount = value; }
```

Berikut adalah data member, constructor dan property dari class Loan.

B. Create loan

Untuk Memenuhi Kebutuhan *create* kami membuat beberapa method sebagai berikut :

- AddData

AddData

Proses pengajuan hutang akan menggunakan method add data. Method ini akan dipanggil di form tambah loans.

```
public static void AddData(Loan loan)// menambah data loan yang baru ke tabel loans
{
    string sql = "insert into loans(idloans,lender_account_number,amount,due_date) " +
    "values(" + loan.IdLoan + ", '" + loan.Lender.AccountNumber + "' , " + loan.amount + ",'" + loan.DueDate.ToString("yyyy-MM-dd hh:mm:ss")+"')";
    Connection.ExecuteNonQuery(sql);
}
```

C. Read Loan

Untuk Memenuhi Kebutuhan *read* kami membuat beberapa method sebagai berikut :

- BacaData
- FetchDataByLender
- LoanCheck

BacaData

Agar karyawan dapat mengakses data pengajuan hutang dari user yang banyak. Maka akan dibutuhkan *method* baca data. Fungsi dari method ini adalah membaca data dari entitas loans yang ada di dalam database. *Method* ini hanya dipanggil di dalam form daftar loan. Hanya karyawan diba yang

mampu mengakses form daftar loan. *Method* BacaData memiliki 2 varian. Varian pertama membaca *column* yang *null* sedangkan varian lain membaca kriteria yang diberikan.

```

1 reference
public static List<Loan> BacaData(string kriteria)//membaca column yang null
{
    string sql;
    if (kriteria == "")
    {
        sql = "select * from loans ";
    }
    else
    {
        sql = "select * from loans " +
            "where " + kriteria + " is null ";
    }

    MySqlDataReader hasil = Connection.ExecuteReader(sql);
    List<Loan> listloan = new List<Loan>();
    while (hasil.Read() == true)
    {
        Employee verifikator = null;
        Saving lender = Saving.FetchByID(hasil.GetValue(1).ToString());
        Loan hutang = new Loan(int.Parse(hasil.GetValue(0).ToString()), lender, verifikator, double.Parse(hasil.GetValue(3).ToString()), DateTime.Parse(hasil.GetValue(4).ToString()));
        listloan.Add(hutang);
    }

    return listloan;
}

0 reference
public static List<Loan> BacaData(string kriteria, string nilaiKriteria)// untuk membaca dengan kriteria lain
{
    string sql;
    if (kriteria == "")
    {
        sql = "select * from loans ";
    }
    else
    {
        sql = "select * from loans " +
            "where " + kriteria + " like '%" + nilaiKriteria + "%'";
    }

    MySqlDataReader hasil = Connection.ExecuteReader(sql);
    List<Loan> listloan = new List<Loan>();
    while (hasil.Read() == true)
    {
        Employee verifikator;
        if (hasil.GetValue(2).ToString() == "")
        {
            verifikator = null;
        }
        else
        {
            verifikator = Employee.FetchDataByID(int.Parse(hasil.GetValue(2).ToString()));
        }
        Saving lender = Saving.FetchByID(hasil.GetValue(1).ToString());
        Loan hutang = new Loan(int.Parse(hasil.GetValue(0).ToString()), lender, verifikator, double.Parse(hasil.GetValue(3).ToString()), DateTime.Parse(hasil.GetValue(4).ToString()));
        listloan.Add(hutang);
    }

    return listloan;
}

```

Berikut adalah *code* dari *method* BacaData.

FetchDataByLender

```

3 reference
public static Loan FetchDataByLender(Saving saving)// mengambil 1 objek loan dari tabel loans
{
    string sql = "select * from loans " +
        "where lender_account_number = '" + saving.AccountNumber + "' ";
    MySqlDataReader hasil = Connection.ExecuteReader(sql);
    if(hasil.Read()== true && hasil.GetValue(1).ToString() != "")
    {
        Saving lender = Saving.FetchByID(hasil.GetValue(1).ToString());
        Employee verifikator;
        if (hasil.GetValue(2).ToString() == "")
        {
            verifikator = null;
        }
        else
        {
            verifikator = Employee.FetchDataByID(int.Parse(hasil.GetValue(2).ToString()));

            Loan hutang = new Loan(int.Parse(hasil.GetValue(0).ToString()), lender, verifikator, double.Parse(hasil.GetValue(3).ToString()),DateTime.Parse(hasil.GetValue(4).ToString()));
            return hutang;
        }
    }
    else
    {
        return null;
    }
}

```

Method ini mengambil data berdasarkan data tabungan peminjam.

LoanCheck

```
1 reference
public static bool LoanCheck(Saving saving)
{
    string command = $"SELECT * FROM loans WHERE lender_account_number = '{saving.AccountNumber}'";
    MySqlDataReader hasil = Connection.ExecuteQuery(command);

    if (hasil.Read())
    {
        //Mempunyai loan
        return true;
    }
    else
    {
        //Tidak mempunyai loan
        return false;
    }
}
```

Method ini akan melakukan pengecekan apakah tabungan pengguna memiliki hutang atau tidak.

D. Update Loan

Untuk Memenuhi Kebutuhan *update* kami membuat beberapa method sebagai berikut :

- UpdateData

UpdateData

```
1 reference
public static void UpdateData(Loan loan)
{
    string sql = $"UPDATE loans SET lender_account_number = '{loan.Lender.AccountNumber}', verifikator = {loan.Verifikator.IdEmployee}, amount = {loan.Amount} WHERE idloans = {loan.IdLoan}";
    Connection.ExecuteNonQuery(sql);
}
```

Method update data berfungsi untuk mengubah data loan yang ada di dalam database. Method ini akan dipanggil ketika karyawan melakukan konfirmasi pengajuan hutang. Ketika karyawan melakukan konfirmasi pengajuan hutang , id dari karyawan tersebut akan otomatis masuk ke dalam database sebagai verifikator dari pengajuan hutang yang di konfirmasi.

E. Delete Loan

Untuk Memenuhi Kebutuhan *delete* kami membuat beberapa method sebagai berikut :

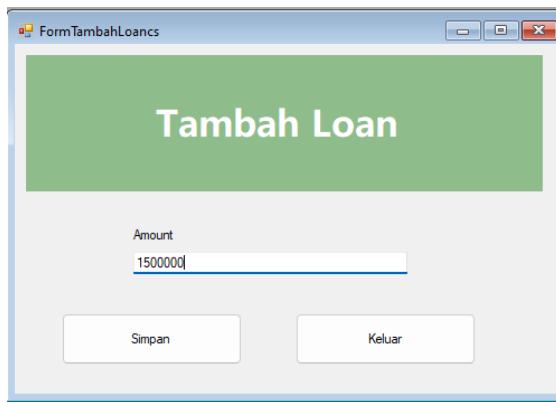
- DeleteData

DeleteData

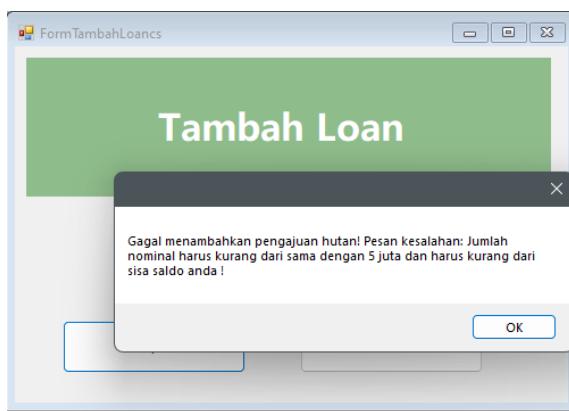
```
1 reference
public static bool DeleteLoan(Loan loan)//menghapus data loan , dipanggil ketika pengguna melakukan pembayaran hutang
{
    string command = $"DELETE FROM loans WHERE idloans = {loan.IdLoan}";
    int rowAffected = Connection.ExecuteNonQuery(command);
    if (rowAffected == 0)
    {
        return false;
    }
    else
    {
        return true;
    }
}
```

Method delete data akan berfungsi untuk menghapus data hutang yang telah dibayar. Setelah itu pengguna dapat membuat pengajuan hutang kembali.

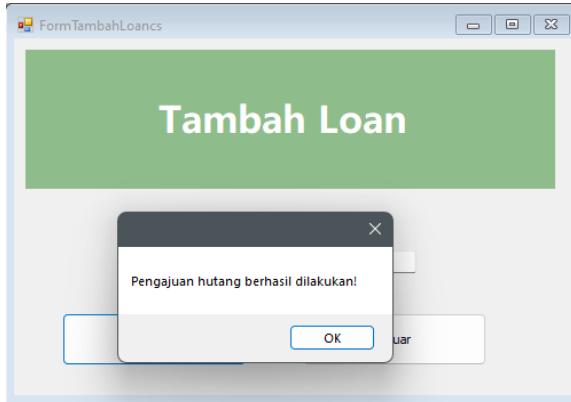
F. Pengajuan Hutang



Pengguna dapat membuat pengajuan hutang melalui FormTambahLoan. Pengguna harus memastikan hutang yang ingin diajukan kurang dari 5 juta dan kurang dari sisa saldo tabungan. Jika tidak program akan mengirim pesan *exception*. FormTambahLoan dipanggil melalui *menu strip* hutang.



Pengguna akan berhasil membuat pengajuan hutang jika hutang kurang dari 5 juta dan kurang dari sisa saldo tabungan.



```

private void buttonSimpan_Click(object sender, EventArgs e)
{
    try
    {
        if (double.Parse(textBoxAmount.Text) <= 5000000 && double.Parse(textBoxAmount.Text) < formMain.saving.Balance)
        {
            Loan newLoan = new Loan(Loan.GenerateID(), formMain.saving, new Employee(), double.Parse(textBoxAmount.Text), DateTime.Now.AddMonths(6));
            Loan.AddData(newLoan);
            MessageBox.Show("Pengajuan hutang berhasil dilakukan!");
        }
        else
        {
            throw (new ArgumentException("Jumlah nominal harus kurang dari sama dengan 5 juta dan harus kurang dari sisa saldo anda !"));
        }
    }
    catch(Exception exception)
    {
        MessageBox.Show($"Gagal menambahkan pengajuan hutang! Pesan Kesalahan: {exception.Message}");
    }
}

```

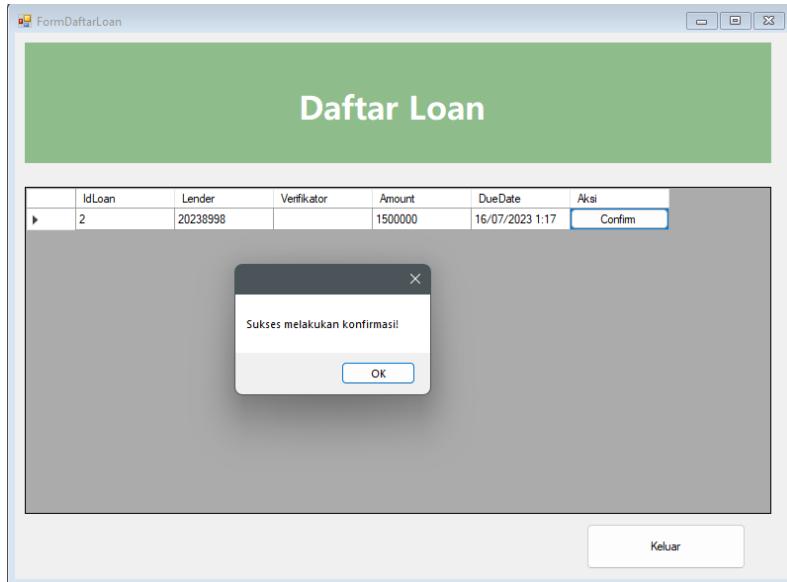
Berikut adalah *code* yang dijalankan ketika pengguna membuat pengajuan hutang.

G. Konfirmasi Hutang

	IdLoan	Lender	Verifikator	Amount	DueDate	Aksi
▶	2	20238998		1500000	16/07/2023 1:17	<input type="button" value="Confirm"/>

Hanya pegawai DiBa saja yang memiliki akses untuk mengkonfirmasi pengajuan hutang pengguna. Pegawai dapat melihat daftar hutang yang belum dikonfirmasi melalui FormDaftarHutang, data hutang yang belum dikonfirmasi dapat dilihat dengan data verifikator yang masih null. Untuk melakukan konfirmasi pegawai dapat mengklik *button confirm* di baris data hutang yang ingin

dikonfirmasi. Dengan mengklik *button confirm*, rekening pemilik hutang akan otomatis mendapat uang tambahan senilai nominal hutang yang dibuat. Hutang yang dikonfirmasi juga akan masuk ke dalam riwayat transaksi si pemilik hutang.



```

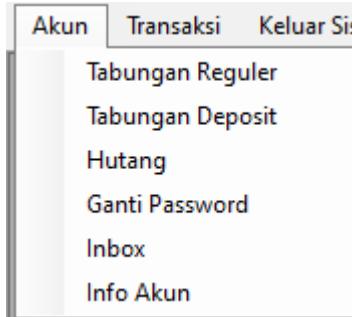
private void dataGridViewLoan_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    if (formMain.user == null)
    {
        if (e.ColumnIndex == dataGridViewLoan.Columns["btnConfirmGrid"].Index && e.RowIndex >= 0)
        {
            try
            {
                Loan selectedLoan = (Loan)dataGridViewLoan.CurrentRow.DataBoundItem;
                if (selectedLoan.Verifikator == null)
                {
                    selectedLoan.Verifikator = formMain.employee;
                    Loan.UpdateData(selectedLoan);

                    Transaction transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now, selectedLoan.Amount, "Menambah Loan"
                        , Saving.FetchByID("00226789"), selectedLoan.Lender, TransactionType.FetchDataByID(1));
                    Transaction.TambahTransaksi(transaksi, selectedLoan, "konfirmLoan");
                    Inbox inbox = new Inbox(Inbox.GenerateID(), Inbox.GenerateInbox(selectedLoan, "konfirmLoan"), DateTime.Now, "Unread",
                        DateTime.Now, User.FetchDataByID(selectedLoan.Lender.User.Nik));
                    Inbox.AddData(inbox);
                    MessageBox.Show($"Sukses melakukan konfirmasi!");
                    FormDaftarLoan_Load(buttonKeluar, e);
                }
            }
            catch (Exception exception)
            {
                MessageBox.Show($"Terjadi kesalahan! Pesan kesalahan: {exception.Message}");
            }
        }
    }
}

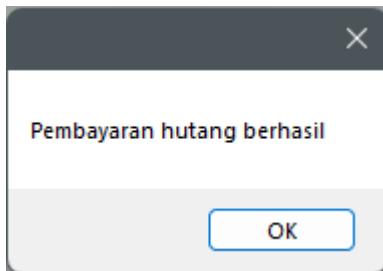
```

Berikut adalah *code* yang dijalankan ketika pegawai melakukan konfirmasi hutang.

H. Pembayaran Hutang



Untuk melakukan pembayaran hutang, pengguna dapat melalui *menu strip* hutang. Jika hutang pengguna belum di konfirmasi oleh pegawai, program akan mengeluarkan pesan bahwa hutang belum di konfirmasi. Jika sudah, program akan mengeluarkan *message box* berisi nominal hutang dan apakah pengguna bersedia melakukan pembayaran hutang.



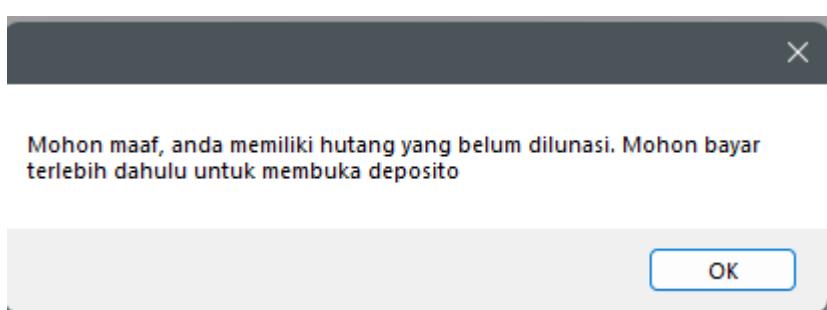
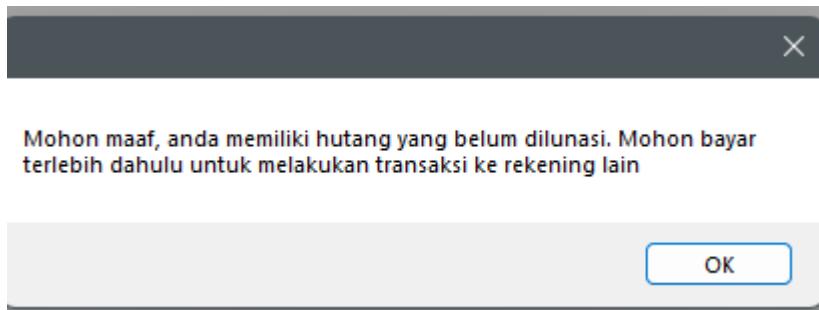
Setelah pembayaran hutang berhasil, saldo rekening pengguna akan berkurang sesuai dengan nominal hutang yang dibayar dan akan dialihkan ke saldo tabungan khusus DiBa.

```
private void hutangToolStripMenuItem_Click(object sender, EventArgs e)
{
    if ((user == null))
    {
        if (Loan.LoanCheck(saving))
        {
            if (Loan.FetchDataByLender(saving).Verifikator == null)//jika verifikator null maka , hutang belum di konfirmasi
            {
                MessageBox.Show($"Hutang masih dalam proses konfirmasi!");
            }
            else
            {
                DialogResult result = MessageBox.Show($"Sisa hutang: {Loan.FetchDataByLender(saving).Amount.ToString("C0", new CultureInfo("id"))} , apakah anda ingin bayar sekarang ?", "konfirmasi",
                    MessageBoxButtons.YesNo);
                if(result == DialogResult.Yes)
                {
                    Loan loan = Loan.FetchDataByLender(saving);
                    Transaction transaksi = new Transaction(Transaction.GenerateId(), DateTime.Now, loan.Amount, "Bayar Loan", saving,
                        Saving.FetchById("20226789"), Transactiontype.FetchDataById(1));
                    Transaction.TambahTransaksi(transaksi, loan, "bayarLoan");
                    Inbox inbox = new Inbox(inbox.GenerateID(), inbox.GenerateInbox(loan, "bayarLoan"), DateTime.Now, "Unread",
                        DateTime.Now, User.FetchDataByID(user.NIK));
                    inbox.AddData(inbox); // mengirim pesan ke inbox pengguna , bahwa hutang telah terbayar
                    Loan.DeleteLoan(loan); //menghapus data hutang pengguna di tabel loans, supaya pengguna bisa membuat pengajuan hutang di masa depan.
                    MessageBox.Show("Pembayaran hutang berhasil");
                }
            }
        }
        else
        {
            Form form = Application.OpenForms["FormTambahLoans"];
            if (form == null)
            {
                FormTambahLoans formTambahLoans = new FormTambahLoans();
                formTambahLoans.MdiParent = this;
                formTambahLoans.Show();
            }
            else
            {
                form.Show();
                form.BringToFront();
            }
        }
    }
}
```

Berikut adalah *code* yang dijalankan ketika pengguna membayar hutang.

I. Konsekuensi telat membayar hutang.

Jika pengguna telat membayar hutang, pengguna tidak akan bisa melakukan proses transaksi dengan sesama pengguna DiBa dan tidak bisa membuka deposito.



```
//Melakukan pengecekan tanggal ketika pengguna login aplikasi. Bagi tabungan yang memiliki loans
2 references
public static bool DueDateCheck(Saving saving)
{
    string sql = $"select due_date from loans " +
        $" where lender_account_number = '{saving.AccountNumber}'";
    MySqlDataReader hasil = Connection.ExecuteQuery(sql);

    if(hasil.Read() == true && hasil.GetValue(0).ToString() != "")
    {
        if (DateTime.Now < DateTime.Parse(hasil.GetValue(0).ToString()))
        {
            return false;
        }
        else
        {
            return true;
        }
    }
    else
    {
        return false;
    }
}
```

Method DueDataCheck akan mengeluarkan nilai *false* jika pengguna tidak memiliki hutang atau pengguna tidak melewati waktu tenggat pembayaran. Method ini akan mengeluarkan nilai *true* jika pengguna melewati waktu tenggat pembayaran. Method ini akan dipanggil ketika pengguna mengklik *menu strip* deposit dan *menu strip* transaksi.

Kesulitan

Kesulitan yang kami alami selama penggerjaan project antara lain :

- Sulitnya membagi waktu dikarenakan adanya project dari mata kuliah lain.
- Menemukan error yang penyebab dan solusinya sulit ditemukan.

Dokumentasi

Tanggal: 17 Desember 2022

Topik: Pembuatan User Interface & Penerapan Class ke Form

Deskripsi:

Membuat desain form sebagai *user interface* dari aplikasi DiBa. Daftar form yang dibuat, yaitu FormAddressBook, FormBacaInbox, FormCekPin, FormChangePassword, FormDaftarDeposito, FormDaftarInbox, FormDaftarJenisTransaksi, FormDaftarLoan, FormDaftarMerchants, FormDaftarPegawai, FormDaftarPosition, FormDaftarTabunganReguler, FormGantiPin, FormInfoAkun, FormLogin, FormMain, FormPengajuanPencairan, FormRegister, FormRiwayatTransaksi, FormTabunganReguler, FormTambahAddressBook, FormTambahDeposito, FormTambahInbox, FormTambahJenisTransaksi, FormTambahLoans, FormTambahMerchants, FormTambahPegawai, FormTambahPin, FormTambahPosition, FormTambahTransaksi, FormTopUpSaldo, FormUbahAddressBook, FormUbahJenisTransaksi, FormUbahMerchant, FormUbahPegawai, FormUbahPosition.

Selain membuat desain form, dilakukan juga proses penerapan *class* ke *form*. Daftar *class* yang diterapkan adalah AddressBook, Deposito, Transation, Loan, dan Merchant.

Foto:

