COMARCH ERP

Integracja Comarch e-Sprawozdania

Comarch
e-Sprawozdania

Spis treści

1 Repozytorium danych

1.1 Konektor

Aplikacja **Comarch e-Sprawozdania** może zostać oparta o repozytorium, gdzie będzie przechowywane sprawozdanie finansowe oraz jego załączniki. W dalszym ciągu tej instrukcji będzie mowa o tabelach bazy danych, chociaż źródło danych może być dowolne.

Biblioteka odpowiedzialna za zarządzanie integracją, czyli odczytem i zapisem odpowiednich informacji z repozytorium danych, w dalszej części instrukcji będzie umownie nazywana **Konektorem**.

1.2 Struktura bazy danych

Miejsce gdzie przechowywane będą odpowiednie informacje pochodzące z aplikacji Comarch e-Sprawozdania powinno mieć odpowiednią strukturę. W przypadku tabeli bazy danych powinna ona wyglądać następująco:

1.2.1 Tabela główna sprawozdania

Tabela główna sprawozdania przechowuje podstawowe informacje o sprawozdaniu finansowym jak i samą jego treść **PlikXML**. Zawiera także plik konfiguracyjny **PlikKonfigXML** przechowujący ustawienia parametrów sprawozdania.

`Id [int] IDENTITY(1,1) NOT NULL,`	Identyfikator
`NazwaFirmy [varchar](3500) NULL,`	Nazwa firmy
`DataDo [int] NULL,`	Data do kiedy okres sprawozdania
`DataSporzadzenia [int] NULL,`	Data sporządzenia
`Typ [int] NULL,`	Typ sprawozdania
`WersjaSchemy [varchar](10) NULL,`	Wersja schemy
`Status [int] NULL,`	Status
`Aktywny [int] NULL,`	Czy dokument jest aktywny
`PlikXml [varbinary](max) NULL,`	Plik XML zawierający dane sprawozdania
`PlikKonfigXm [varbinary](max) NULL,`	Plik konfiguracyjny w postaci XML

1.2.2 Tabela załączników sprawozdania

Ze względów wydajnościowych załączniki do sprawozdania finansowe w postaci m.in. dokumentów PDF przechowywane są w oddzielnej tabeli.

`Id [int] IDENTITY(1,1) NOT NULL,`	Identyfikator
`SFNId [int] NULL,`	Identyfikator sprawozdania finansowego
`Nazwa [varchar](400) NULL,`	Nazwa pliku
`Plik [varbinary](max) NULL,`	Plik
`Typ [int] NULL,`	Typ dokumentu

1.3 Mapowanie struktury na kod .NET

Przedstawione wyżej tabele mapowane są na następujące klasy w interfejsie konektora

1.3.1 Tabela główna sprawozdania

```
public class FinancialStatementBasic
{
    public int Id -> [Id]
    public string CompanyName -> [NazwaFirmy]
    public int Active -> [Aktywny]
    public DateTime DateTo -> [DataDo]
    public DateTime PreparationDate -> [DataSporzadzenia]
    public FinancialStatementType FinancialStatementType -> [Typ]
    public FinancialStatementStatus Status - >[Status]
}

public class FinancialStatement : FinancialStatementBasic
{
    public string FileXml - > [PlikXml]
    public string FileXmlConfig -> [PlikKonfigXml]
    public string Version -> [WersjaSchemy]
}
```

1.3.2 Tabela załączników sprawozdania

```
public class Attachment : AttachmentBasic
{
    public byte[] File -> [Plik]
}

public class AttachmentBasic
{
    public int Id  -> [Id]
    public int IdFinancialStatementId -> [SFNId]
    public string FileName -> [Nazwa]
    public DocumentTypeAttachment DocumentType -> [Typ]
}
```

2 Lista wykorzystywanych metod

2.1 Specyfikacja interfejsu

Wszystkie metody wywoływane przez aplikację Comarch e-Sprawozdania należą do intefejsu IFinancialStatementRepository_2019_1_0, który należy zaimplementować. Interfejs i definicja poszczególnych klas znajdują się w bibliotece ESprawozdania.Integration.dll

2.2 Wykorzystywane metody

2.2.1 Logowanie

```
//Metoda do logowania.
bool CheckUser(Database database, Company company, User user, string
password);

//Metoda to weryfikacji NIP-u.
bool CheckTaxIdentificationNumber(string taxIdentificationNumberKey);

//Pobranie klucza z licencją, nie jest konieczne do implementacji.
string GetKeyServer();

//Wylogowanie z konektora.
bool Logout();
```

2.2.2 Zarządzanie uprawnieniami użytkowników

```
//Pobieranie listy dostępnych baz.
IList<Database> GetDatebases();

//Pobieranie listy dostępnych baz na podstawie wybranego użytkownika.
IList<Database> GetDatebases(User user);

//Pobieranie listy użytkowników.
IList<User> GetUsers();

//Pobieranie listy użytkowników na podstawie wybranej bazy danych.
IList<User> GetUsers(Database database);

//Pobranie nazwy komponentu który zostanie wyświetlony na liście do wyboru.
string GetComponentName();

//Pobranie typu logowania, zostały udostępnione trzy sposoby logowania,
dostępne są do wyboru LoginType.DatabaseUser, LoginType.UserDatabase,
LoginType.DatabaseUserDatabase
LoginType GetLoginType();

//Pobieranie listy firm na podstawie bazy danych.
IList<Company> GetCompanies(Database database);
```

2.2.3 Pobranie sprawozdania

```
//Dodawanie sprawozdania finansowego.
int InsertFinancialStatement(FinancialStatement FinancialStatement);

//Aktualizacja sprawozdania finansowego.
bool UpdateFinancialStatement(FinancialStatement FinancialStatement);

//Usunięcie sprawozdania finansowego.
```

```
bool DeleteFinancialStatement(int idFinancialStatement);

//Pobranie sprawozdania finasowego.
FinancialStatement GetFinancialStatement(int idFinancialStatement);

//Lista wszystkich sprawozdań finansowych(Obiekt nie zawiera wszystkich pól
tabeli). Dane są wyświetlane na liście sprawozdań finasowych.
IList<FinancialStatementBasic> GetFinancialStatements();
```

2.2.4 Pobranie załącznika

```
//Pobranie pliku z tabeli z załącznikami, w przykładzie jest to pole
„Plik”.
byte[] GetAttachmentFile(int idAttachment);

//Usunięcie załącznika do sprawozdania finansowego.
bool DeleteAttachment(int idAttachment);

//Dodanie załącznika do sprawozdania finasowego.
int InsertAttachment(Attachment attachment);

//Aktualizacja załącznika do sprawozdania fiansowego.
bool UpdateAttachment(Attachment attachment);
```

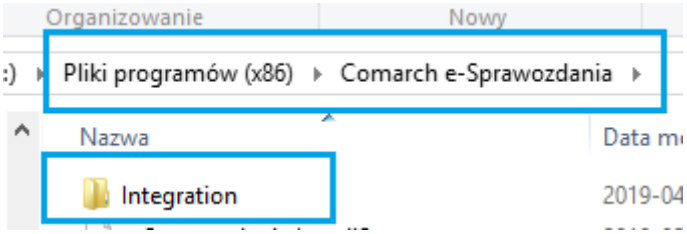
2.2.5 Zestawienia księgowe

```
//Pobranie listy zestawień finansowych które można importować do
sprawozdania finansowego. Dane są wyświetlane na liście zestawień.
IList<FinancialReport> GetFinancialReports();

//Pobranie zestawienia finansowego które można importować do sprawozdania
finansowego.
string GetFinancialReport(int idFinancialReport, bool calculate, DateTime
periodFrom, DateTime periodTo);
```

2.3 Instalacja biblioteki

Bibliotekę DLL z z zaimplementowanym interfejsem należy umieścić w katalogu „Integration”, w miejscu gdzie są zainstalowane e-Sprawozdania



Rys 1. Instalacja nowej biblioteki

3 Przykładowy kod

Poniżej przykładowa implementacja interfejsu konektora

```
public bool CheckTaxIdentificationNumber(string taxIdentificationNumberKey)
{
    if (GetFirmaRoot().Frm_NIP == taxIdentificationNumberKey)
    {
        return true;
    }
    else
    {
        return false;
    }
}

public bool CheckUser(Database database, Company company, User user, string
password)
{
    DatabaseManager databaseManager = new DatabaseManager();
    string decryptfrowanaValue =
encryptionManager.Decrypt(databaseManager.GetConnectionString(database.Name
));
    connectionString = decryptfrowanaValue.Split(':')[1];
    if (connectionString == "")
    {
        return false;
    }
    string hash = String.Empty;
    OpeKarty opeKarty;
    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        opeKarty = dbcontext.OpeKarty.Single(ope => ope.Ope_GIDNumer ==
user.Id);
    }

    using (SHA256 passwordSHA256 = SHA256.Create())
    {
        byte[] crypto =
passwordSHA256.ComputeHash(Encoding.ASCII.GetBytes(password));
        foreach (byte theByte in crypto)
        {
            hash += theByte.ToString("x2");
        }
    }
    hash = hash.ToUpper();

    if (opeKarty.Ope_Haslo == hash)
    {
        return true;
    }
    else
```

```
    {
        connectionString = "";
        return false;
    }

}

public bool DeleteAttachment(int idAttachment)
{
    bool wynik = false;

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFZalaczniki zalacznik = dbcontext.SFZalaczniki.Single(r =>
r.SFZ_Id == idAttachment);

        dbcontext.SFZalaczniki.Remove(zalacznik);
        wynik = dbcontext.SaveChanges() == 0 ? false : true;
        dbcontext.Dispose();
    }

    return wynik;

}

public bool DeleteFinancialStatement(int idFinancialStatement)
{
    bool wynik = false;

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFNag naglowekSF = dbcontext.SFNag.Single(r => r.SFN_Id ==
idFinancialStatement);

        dbcontext.SFNag.Remove(naglowekSF);
        wynik = dbcontext.SaveChanges() == 0 ? false : true;
        dbcontext.Dispose();
    }

    return wynik;
}

public byte[] GetAttachmentFile(int idAttachment)
{
    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFZalaczniki zalacznik = dbcontext.SFZalaczniki.Single(r =>
r.SFZ_Id == idAttachment);

        return zalacznik.SFZ_Plik;
    }
}
```

```
public IList<Company> GetCompanies(Database database)
{
    List<Company> companies = new List<Company>();
    Firma firma = GetFirmaRoot(database);

    Company company = new Company()
    {
        Name = firma.Frm_Nazwa1
    };
    companies.Add(company);
    return companies;
}

public string GetComponentName()
{
    return "KONEKTOR PRZYKLADOWY";
}

public IList<Database> GetDatebases()
{
    List<Database> databases = new List<Database>();
    DatabaseManager databaseManager = new DatabaseManager();
    foreach (string datebaseName in
databaseManager.GetDatabaseNamesLists())
    {
        databases.Add(new Database() { Name = datebaseName });
    }

    return databases;
}
        }

        public IList<Database> GetDatebases(User user)
{
    List<Database> databases = new List<Database>();
    DatabaseManager databaseManager = new DatabaseManager();
    foreach (string datebaseName in
databaseManager.GetDatabaseNamesLists())
    {
        databases.Add(new Database() { Name = datebaseName });
    }

    return databases;
}

public string GetFinancialReport(int idFinancialReport, bool calculate,
DateTime periodFrom, DateTime periodTo)
{
    FinancialReportService frs = new
FinancialReportService(connectionString);
    return frs.GetFinancialReport(idFinancialReport, calculate, periodFrom,
periodTo);
}
```

```
public IList<FinancialReport> GetFinancialReports()
{
    FinancialReportService frs = new
FinancialReportService(connectionString);
    return frs.GetFinancialReports();
}

public FinancialStatement GetFinancialStatement(int idFinancialStatement)
{
    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFNag naglowekSF = dbcontext.SFNag.Single(r => r.SFN_Id ==
idFinancialStatement);

        FinancialStatement financialStatement =
CreateFinancialStatementFromSFNag(naglowekSF);
        financialStatement.AttachmentBasic = new List<AttachmentBasic>();
        foreach (var attachmentB in dbcontext.SFZalaczniki.Select(zal =>
new { zal.SFZ_Id, zal.SFZ_SFNId, zal.SFZ_Nazwa, zal.SFZ_Typ }).Where(z =>
z.SFZ_SFNId == idFinancialStatement).ToList())
        {
            AttachmentBasic attachmentBasic = new AttachmentBasic();
            attachmentBasic.Id = attachmentB.SFZ_Id;
            attachmentBasic.IdFinancialStatementId = idFinancialStatement;
            attachmentBasic.FileName = attachmentB.SFZ_Nazwa;
            attachmentBasic.DocumentType =
(DocumentTypeAttachment)attachmentB.SFZ_Typ;
            financialStatement.AttachmentBasic.Add(attachmentBasic);
        }
        return financialStatement;
    }
}

public IList<FinancialStatementBasic> GetFinancialStatements()
{
    List<FinancialStatementBasic> financialStatements = new
List<FinancialStatementBasic>();

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        foreach (SFNag row in dbcontext.SFNag.ToList())
        {

financialStatements.Add(CreateFinancialStatementBasicFromSFNag(row));
        }
    }

    return financialStatements;
}

public LoginType GetLoginType()
{
```

```
    return LoginType.DatabaseUser;
}

public IList<User> GetUsers()
{
    return new List<User>();
}

public IList<User> GetUsers(Database database)
{
    List<User> users = new List<User>();
    using (Entities dbcontext = new
Entities(GetEntityConnectionString(database)))
    {
        foreach (OpeKarty row in dbcontext.OpeKarty.Where(ope =>
ope.Ope_EsprawozdaniaFinansowe == 1 && ope.Ope_Zablokowane < 1).ToList())
        {
            users.Add(ConvertOpeKartyToUser(row));
        }
    }
    return users;
}

public int InsertAttachment(Attachment attachment)
{
    SFZalaczniki zalacznik = new SFZalaczniki();
    MapAttachmentToSFZalaczniki(ref zalacznik, attachment);

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        dbcontext.SFZalaczniki.Add(zalacznik);
        dbcontext.SaveChanges();
        dbcontext.Dispose();
    }

    attachment.Id = zalacznik.SFZ_Id;
    return zalacznik.SFZ_Id;
}
        }


        public int InsertFinancialStatement(FinancialStatement
FinancialStatement)
{
    SFNag naglowekSF = new SFNag();
    MapFinancialStatementToSFNag(ref naglowekSF, FinancialStatement);

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        dbcontext.SFNag.Add(naglowekSF);
        dbcontext.SaveChanges();
        dbcontext.Dispose();
    }
```

```
    FinancialStatement.Id = naglowekSF.SFN_Id;
    return naglowekSF.SFN_Id;
}

public bool UpdateAttachment(Attachment attachment)
{
    bool wynik = false;

    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFZalaczniki zalacznik = dbcontext.SFZalaczniki.Single(r =>
r.SFZ_Id == attachment.Id);
        MapAttachmentToSFZalaczniki(ref zalacznik, attachment);

        wynik = dbcontext.SaveChanges() == 0 ? false : true;
        dbcontext.Dispose();
    }

    return wynik;
}


public bool UpdateFinancialStatement(FinancialStatement FinancialStatement)
{
    bool wynik = false;
    using (Entities dbcontext = new Entities(GetEntityConnectionString()))
    {
        SFNag naglowekSF = dbcontext.SFNag.Single(r => r.SFN_Id ==
FinancialStatement.Id);
        MapFinancialStatementToSFNag(ref naglowekSF, FinancialStatement);

        List<int> attachmentDatabaseIdList =
dbcontext.SFZalaczniki.Select(zal => new { Id = zal.SFZ_Id, SFNId =
zal.SFZ_SFNId }).Where(z => z.SFNId == FinancialStatement.Id).Select(s =>
s.Id).ToList();

        List<int> attachmentInterfaceIdList = new List<int>();
        foreach (AttachmentBasic attachmentB in
FinancialStatement.AttachmentBasic)
        {
            attachmentInterfaceIdList.Add(attachmentB.Id);
        }

        foreach (int id in attachmentDatabaseIdList)
        {
            if (!attachmentInterfaceIdList.Contains(id))
            {
                SFZalaczniki sFZalaczniki = dbcontext.SFZalaczniki.Single(z
=> z.SFZ_Id == id);
                dbcontext.SFZalaczniki.Remove(sFZalaczniki);
            }
        }
```

```
            wynik = dbcontext.SaveChanges() == 0 ? false : true;
        }
        return wynik;
    }

    public bool Logout()
    {
        return true;
    }

    public string GetKeyServer()
    {
        return "";
    }
```