Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

**Objektno oblikovanje**

Seminar za Laboratorij profila PIIS

*OR mapiranje primijenjeno u „Sustavu za programsku podršku knjigovodstvu udruga“*

Juraj Dončević (0036463062)

Matej Vukosav (0036473765)

Zagreb, 16.1.2017.

# Design story

# Use cases

# Perzistencija kroz O/R mapiranje

Zajednički izvor podataka biti će u obliku SQLite baze podataka. Baza podataka će biti sadržana u jednoj SQLite datoteci.

Zajednički repozitorij podataka će stvarati vezu prema bazi podataka, te će vršiti O/R mapiranje podataka iz baze. O/R mapiranje će biti izvršeno NHibernate značajkom. Na taj način će zajednički repozitorij nuditi i CRUD metode, te još neke prilagođene metode dohvata. Repozitorij će također sadržavati osnovnu definiciju modela podataka koji će se koristiti u cijelom sustavu.

Perzistencija se provodi koristeći O/R mapiranje s Fluent Nhibernateom u SQLite bazu podataka.

Fluent Nhibernate za razliku od običnog Nhibernatea ne provodi mapiranje putem XML-a već unutar C# koda. Predefiniranim setom naredbi objekti baze se prevode u objekte koji se mogu koristiti unutar aplikacije.

UserMap je model koji mapira User klasu.

public class UserMap : ClassMap<User>

{

public UserMap()

{

Table("User");

Id(x => x.Id).GeneratedBy.Native().Unique();

Map(x => x.Username).Not.Nullable();

Map(x => x.Password).Not.Nullable();

Map(x => x.Email).Nullable();

Map(x => x.Address).Nullable();

Map(x => x.AssociationName).Nullable();

Map(x => x.OIB).Nullable();

}

}

InvoiceMap je model koji mapira Invoice klasu.

public class InvoiceMap : ClassMap<Invoice>

{

public InvoiceMap()

{

Id(x => x.Id).GeneratedBy.Increment();

Map(x => x.Date);

Map(x => x.Amount);

Map(x => x.InvoiceClassNumber);

References(c => c.User).Column("FK\_UserId").Not.LazyLoad();

Map(x => x.FK\_UserId).Formula("[FK\_UserId]");

UseUnionSubclassForInheritanceMapping();

}

}

IngoingInvoiceMap i OutgointInvoiceMap su modeli koji mapiraju IngoingInvoice i OutgoingInvoice koristeći zajedničke elemenate iz InvoiceMap modela.

public class IngoingInvoiceMap : SubclassMap<IngoingInvoice>

{

public IngoingInvoiceMap()

{

Map(x => x.SupplierInfo);

}

}

VatMap je klasa koja mapira Vat model.

public class VatMap : ClassMap<Vat>

{

public VatMap()

{

Id(x => x.Id).GeneratedBy.Native().Unique();

Map(x => x.Name).Not.Nullable();

Map(x => x.Percentage).Not.Nullable();

Table("VAT");

}

}

MonetaryFlowMap je klasa koja mapira abstraktnu klasu MonetaryFlow. Sadrži zajedničke elemente od Receipt i Expenditure klase.

public class MonetaryFlowMap : ClassMap<MonetaryFlow>

{

public MonetaryFlowMap()

{

Id(x => x.Id).GeneratedBy.Increment();

Map(x => x.Date);

Map(x => x.AmountCash);

Map(x => x.AmountNonCashBenefit);

Map(x => x.AmountTransferAccount);

References(c => c.Vat).Column("FK\_VAT").Not.LazyLoad();

References(c => c.User).Column("FK\_UserId").Not.LazyLoad();

Map(x => x.FK\_UserId).Formula("[FK\_UserId]");

Map(x => x.FK\_VAT).Formula("[FK\_VAT]");

Map(x => x.JournalEntryNum);

Map(x => x.Total);

UseUnionSubclassForInheritanceMapping();

}

}

ReceiptMap je klasa koja mapira Receipt model.

public class ReceiptMap : SubclassMap<Receipt>

{

}

ExpenditureMap je klasa koja mapira Expenditure model.

public class ExpenditureMap : SubclassMap<Expenditure>

{

public ExpenditureMap()

{

Map(x => x.Article22);

}

}

## Komponente aplikacije

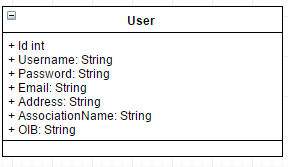
Objekti unutar aplikacije podjeljeni su u nekoliko vrsta prema zadaći koju obavljaju.

### Entitety

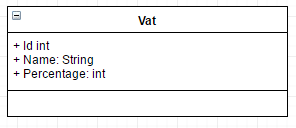
Objekti koji jedinsteno predstavljaju objekt aplikacije. Svaki entitet ima svoj jedinstveni broj koji ga razlikuje od ostalih objekata istog tipa.

Modeli unutar aplikacije su:

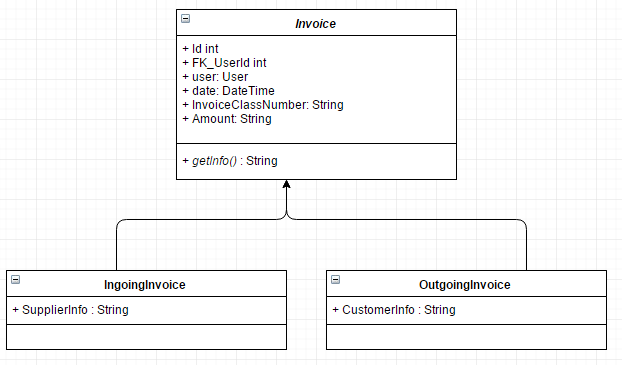
* User je model korisnika aplikacije. Sadrži sve bitne informacije o korisniku unutar aplikacije.



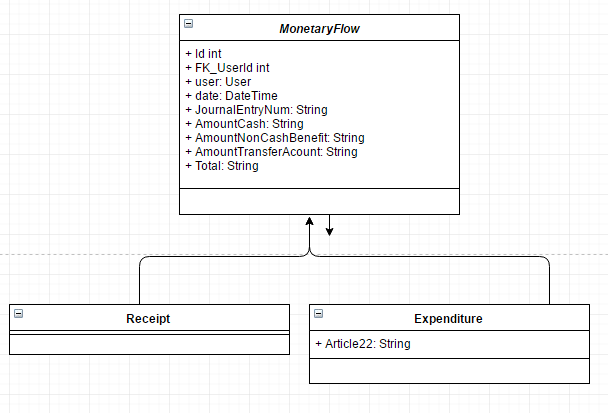
* Vat je model koji predstavlja PDV.



* Invoice – apstraktni model računa koji sadrži zajedničke elemente IngoingInvoice i OutgoingInvoice modela.

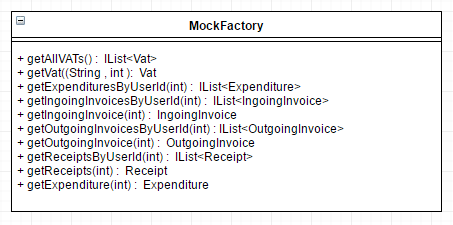


* MonetaryFlow je apstraktni model koji sadrži zajedničke elemente Receipt i Expenditure modela



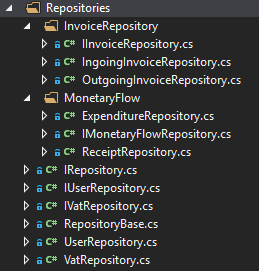
### Tvornice

MockFactory je klasa koja vraća mock objekte aplikacijskih modela.



### Repozitoriji

Repozitoriji su dijelovi aplikacije koji sadrže metode za dohvat objekata iz baze podataka koristeći nHibernate mogučnosti. Postoje repozitoriji za sve navedene entitete. Repozitorij implementira metode navedene u sučelju određenog repozitorija kako bi se omogučila laka zamjena konkretnog repoziorija nekom drugom vrstom implementacije. Postoji 6 repozitorija i 4 sučelja. Repozitoriji su: IngoingInvoiceRepository, OutgointInvoiceRepository, ExpenditureRepository, ReceiptRepository, UserRepository, VatRepository,a sučelja su: IUserRepository, IVatRepository, IMonetaryFlowRepository i IinvoiceRepository



Navest cemo primjer implementacije korisničkog repozitorija i sučelja za korisnički repozitorij.

IUserInterface je sučelje koje sadrži metode koje implementira UserRepository repozitorij.

public interface IUserRepository

{

void Create(User user);

User GetById(int id);

User GetUserByCredentials(UserCredentials userCredentials);

void UpdateUser(User user);

void DeleteUser(User user);

}

public class UserRepository : IuserRepository

ISessionFactory objekt unutar repozitorija je zajednički objekt za sve pozivane metode. Kreira se na početku kreiranja repozitorija i ima istu vrijednost tijekom cijelog korištenja objekta repozitorija. Proizvoljnu implementaciju IsessionFactory objekta moguće je predati objektu preko konstruktora. Ukoliko je konstruktor prazan, koristi se predefinirani ISessionFactory objekt.

private ISessionFactory sessionFactory;

public UserRepository(ISessionFactory sessionFactory)

{

this.sessionFactory = sessionFactory;

}

public UserRepository()

{

this.sessionFactory = SessionManager.SessionFactory;

}

Svaka metoda repozitorija obavlja zadanu radnju koristeći transakcije i sesije.

public void Create(User user)

{

using (var session = sessionFactory.OpenSession())

{

using (ITransaction transaction = session.BeginTransaction())

{

session.SaveOrUpdate(user);

transaction.Commit();

}

}

}