

RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Lietišķo datorzinātņu katedra

Sergejs TERENCEVS
1.kursa RDGDB 1.grupas students
stud. apl. Nr. 061RDB140

VIZUĀLĀ PROGRAMMĒŠANA

Studiju projekta pārskats.

Students:

(paraksts, pārskata iesniegšanas datums)

Atbildīgais mācībspēks:

Gundars Alksnis, docents, Dr.sc.ing.

(paraksts, pārbaudes datums)

Rīga, 2010

Anotācija

Pārskata mērķis ir apkopot informāciju par praktiska uzdevuma – „kursu reģistrēšanas sistēmas” – izpildes gaitā pielietotiem risinājumiem, iegūtam zināšanām un iemaņām.

Pārskats ietver sevī piecas nodaļas. Pirmā nodaļā tiek apskatīta sistēmas datu bāze un projekta struktūra. Otrā nodaļā tiek apskatīts datu abstrakcijas līmenis, atspoguļotas datu struktūras un metodes ar datiem. Trešā nodaļā tiek apskatīta kursu reģistrēšanas darbvirsmas lietojumprogramma, izstrādes gaitā veiktas darbības un formu komponenti. Ceturtā nodaļā tiek apskatīta tīmekļa vietne, izstrādātas tīmekļa ASP.NET formas un to funkcijas. Piektā nodaļa ir paredzēta pārskata izstrādes darbību aprakstām.

Dokumentā ir 20 lappuses, 12 attēli un 1 informācijas avota nosaukums.

Saturs

1.	Sistēmas projekta un datu bāzes apskats	4
1.1.	Veiktās darbības	4
1.2.	Posma secinājumi	5
2.	Datu abstrakcijas līmenis.....	6
2.1.	Veiktās darbības	6
2.2.	Posma secinājumi	8
3.	Darbvirsmas lietojumprogrammas izstrāde	9
3.1.	Veiktās darbības	9
3.2.	Posma secinājumi	10
4.	Tīmekļa vietnes izstrāde	12
4.1.	Veiktās darbības	12
4.2.	Posma secinājumi	15
5.	Pārskata izstrāde.....	16
5.1.	Veiktās darbības	16
5.2.	Posma secinājumi	18
	Nobeigums.....	19
	Literatūra	20

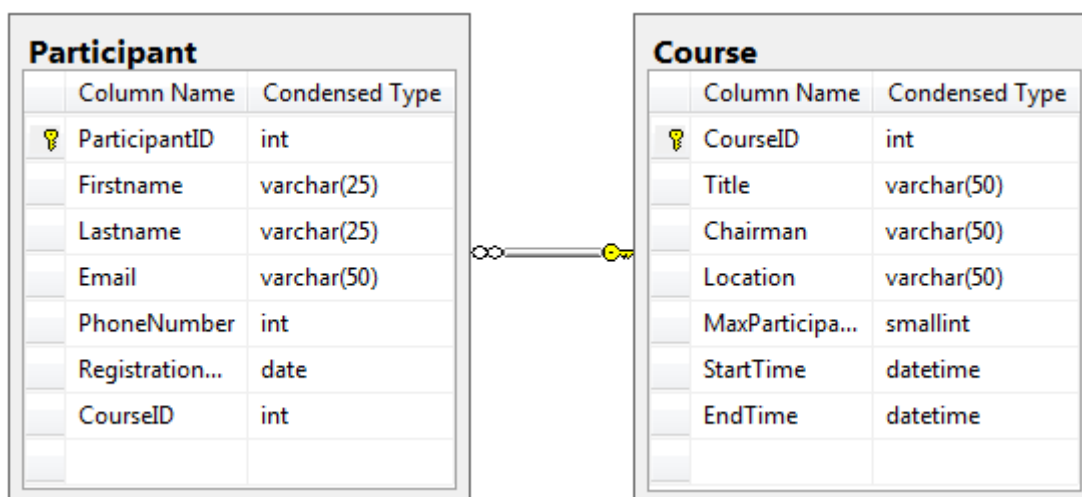
1. SISTĒMAS PROJEKTA UN DATU BĀZES APSKATS

Praktiska darbā tika izstrādāts kursu reģistrēšanas sistēmas prototips. Sistēmas prototips piedāvā iespējas pieteikt jauno kursu, reģistrēties esošam kursam un iegūt pilno priekšstatu par kursa dalībniekiem un norisi, paredzot pilnas informācijas izvadi.

1.1. Veiktās darbības

Sistēmas datu glabāšanai un vadībai tika izvēlēta SQL Server 2008 datu bāzes vadības sistēma. Atbilstoši uzdevuma nosacījumiem tika sastādītas divas tabulas (skat. 1.1.1 attēlu):

- *Course* (kurss) – glabā kursiem raksturīgos datus, piemēram, nosaukums, vadītājs, vieta, laiks un maksimālais dalībnieku skaits;
- *Participant* (dalībnieks) – glabā kursu dalībniekiem raksturīgos datus, piemēram, vārds, uzvārds, e-pasts, tālrunis, reģistrēšanas datums u.tml.



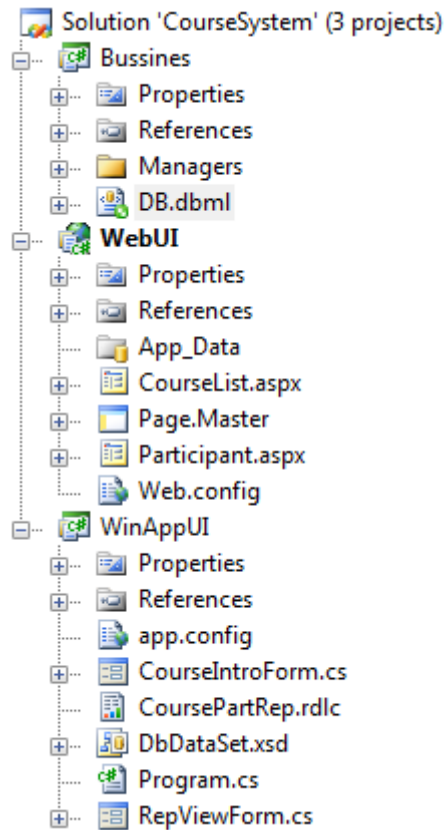
1.1.1 att. Sistēmas datu bāzes tabulas un attiecība starp tām.

Starp datu bāzes tabulām ir paredzēta 1:N relāciju attiecība un atbilstoši uzdevuma nosacījumiem ir sastādīti atbilstošo tabulas lauku datu tipu (skat. 1.1.1 attēlu).

Sistēmas projekts sastāv no vairākiem apakšprojektiem. Katrs apakšprojekts iekapsulē tām būtisko nozīmi un funkcionalitāti. Šādi projekti ir (skat. 1.1.2 projektu):

- *Bussines* (bizness loģika) – reprezentē datu abstrakcijas līmeni, piedāvājot metodes dažādu operāciju veikšanai ar datiem, ka arī ievieš abstrakcijas līmeni divu dažādu paradigmu – objektorientētas un relāciju, datu vienību kartēšanai;
- *WebUI* (tīmekļa lietojuma saskarne) – tīmekļa vietnes projekts, kura piedāvā iespējas reģistrēties kursiem;

- *WinAppUI* (darbvirsmas lietojumprogrammas saskarne) – darbvirsmas lietojumprogrammas un pārskata projekts, kuri ļauj gan pieteikt jauno kursu, gan arī gūt pilno priekšstatu par kursa dalībniekiem un pašu kursu.



1.1.2 att. Sistēmas projekta koks.

1.2. Posma secinājumi

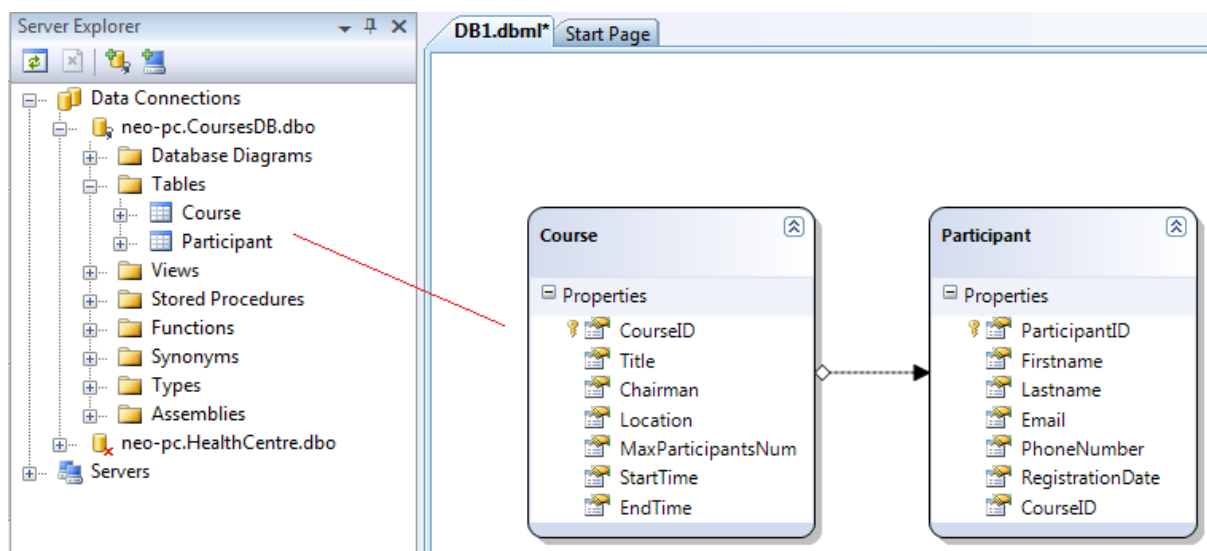
Veicot sistēmas izstrādi, tika novērtēts izstrādes vides Visual Studio ciešs atbalsts esošam kompānijas Microsoft produktiem un tehnoloģijām, piemēram, MS SQL Server, ASP.NET, WPF u.tml., kas ļauj liela mērā samazināt programmatūras izstrādes laiku. Patiecoties šādam atbalstam, .NET platforma tiek izvirzīta par vienu no konkurētspējīgām platformām programmatūras izstrādes jomā.

2. DATU ABSTRAKCIJAS LĪMENIS

Dotajā nodaļā tiek apskatīts projekta datu abstrakcijas līmenis, kurā tiek gan izvērstās vairākas datu struktūras, gan arī ieviestas vairākas metodes datu manipulācijai ar tām.

2.1. Veiktās darbības

Lietojumprogrammas business modeļa projektēšanai tika pielietota LINQ to SQL objekt-relācijas kartēšanas tehnoloģija. Pielietojot dotas tehnoloģijas grafisko konstrukturu, uz esošas datu bāzes shēmas pamatā tika konstruētas 2.1.1 attēlā parādītas klases (entītijas) un attiecīgas asociācijas (saites) starp tām.



2.1.1 att. Entītiju konstruēšana, pielietojot LINQ to SQL grafisko konstrukturu.

Dota konstruktora ģenerētas entītijas tiek izvietotas atsevišķā C# programmēšanas valodas pirmteksta klasē, kurā ar speciālo atribūtu, piemēram, *Column*, *Table*, *Association*, un to atslēgvārdu, piemēram, *DbType*, *CanBeNull*, *IsPrimaryKey*, palīdzību tiek veikta ģenerēto entītiju sasaistīšana (kartēšana) ar datu bāzes tabulām. Dota klase tiek atvasināta no *DataContext* klases, kura nodrošina objektu identitātes un transakciju vadību, seko objektu veiktajām izmaiņām, nodrošina izmaiņu sinhronizēšanu datu bāzes tabulās, piedāvā metodes objektu saglabāšanai, dzēšanai, kā arī to saturu atjaunošanai u.tml [Metha 2008]. Lai darbotos ar datu bāzi vispirms ir nepieciešams izveidot dota konstruktora klases objektu, kā rezultātā tiks atvērts savienojums ar datu bāzi, tiks nodrošinātas visas iepriekšminētas operācijas, kā arī objektu izgūšanas iespējas ar LINQ vaicājumu valodas palīdzību. Kursa dalībnieka entītijas koda fragments izskatās sekojoši:

```

[Table(Name="dbo.Participant")]
public partial class Participant : INotifyPropertyChanging,
    INotifyPropertyChanged {

    private static PropertyChangingEventArgs emptyChangingEventArgs =
        new PropertyChangingEventArgs(String.Empty);

    private int _ParticipantID;
    private string _Firstname;
    private string _Lastname;
    private string _Email;
    private int _PhoneNumber;
    private DateTime _RegistrationDate;
    private int _CourseID;
    private EntityRef<Course> _Course;

    public Participant() {
        this._Course = default(EntityRef<Course>);
        OnCreated();
    }

    [Column(Storage="_ParticipantID", AutoSync=AutoSync.OnInsert,
        DbType="Int NOT NULL IDENTITY", IsPrimaryKey=true,
        IsDbGenerated=true)]
    public int ParticipantID {
        get { return this._ParticipantID; }
        set {
            if ((this._ParticipantID != value)) {
                this.OnParticipantIDChanging(value);
                this.SendPropertyChanging();
                this._ParticipantID = value;
                this.SendPropertyChanged("ParticipantID");
                this.OnParticipantIDChanged();
            }
        }
    }

    ...
}

```

Pēc dotas entitijas pirmkoda var redzēt, ka tā manto *INotifyPropertyChanging* un *INotifyPropertyChanged* interfeisus. Dotie interfeisi kalpo par kontrastu vairākiem notikumiem, piemēram, *PropertyChangingEventHanler*, kuri rodas izmaiņu ieviešana rezultātā un kurus izmanto *DataContext* klase, lai noteiktu izmaiņu sinhronizēšanas nepieciešamību datu bāzes tabulās [Metha 2008].

Datu abstrakcijas līmenī ir paredzētas vairākas metodes, kuras pilda gan nepieciešamo pārbaudi veikšanu, gan pieprasīto datu izgūšanu un objektu ieglābāšanu datu bāzē. Kursa dalībnieka instances ieglābāšanas pirmteksta fragments izskatās šādi:

```

public static void AddParticipant(Participant participant) {
    using (DBDataContext db = new DBDataContext()) {
        db.Participants.InsertOnSubmit(participant);
        db.SubmitChanges();
    }
}

```

Pēc pirmteksta var redzēt, ka objekta ieglabāšanai vispirms tiek izveidots savienojums ar datu bāzi, t.i., izveidota LINQ to SQL konstruktora ģenerētas klases instance, tad objekts tiek pievienots izveidotām kontekstam ar datu bāzi un tiek ieglabāts tajā. Līdzīgi tiek veikta nepieciešamo datu izgūšana, kas tiek panākta ar LINQ paplašinājuma palīdzību. Zemāk ir parādīts pārbaudes piemērs, kuras uzdevums ir noteikt vai reģistrējama kursa dalībnieks nav jau iepriekš reģistrēts.

```
public static bool IsParticipated(int courseID,
    string firstname, string lastname, int phoneNum) {

    using (DBDataContext db = new DBDataContext()) {
        var participant = (from p in db.Participants
            where p.CourseID == courseID &&
                p.Firstname == firstname &&
                p.Lastname == lastname &&
                p.PhoneNumber == phoneNum
            select p).FirstOrDefault();

        return participant != null;
    }
}
```

2.2. Posma secinājumi

Objekt-relāciju kartēšanas tehnoloģijas ļauj novērst vairākas objektorientēto programmēšanas valodu un relāciju datu bāžu integrācijas problēmas, veidojot starp līmeni starp divām dažādam paradigmām un piedāvājot mehānismus doto paradigmu koncepciju pretstatīšanai. Tās pilda klašu hierarhijas saistīšanu ar relāciju datu bāžu tabulām, atbalsta vairākus objektu ielādes un identitātes vadības mehānismus, veic objektu asociāciju kartēšanu attiecībā pret tabulu attiecībām u.tml. Doto tehnoloģiju pielietojums atvieglo programmatūras izstrādes procesu un ļauj ietaupīt izstrādes laiku, ļaujot koncentrēties tieši uz pašu lietojumprogrammas funkcionalitātes izstrādes procesu. Tās piedāvā vairākas objektorientētas vaicājumu valodas, piemēram, HQL, Criteria, LINQ, kuras ļauj darboties ar datu bāzi programmēšanas valodai raksturīga veidā, kas, savukārt, atvieglo sistēmas uzturēšanas procesu un pirmteksta atklūdošanu (*refactoring*).

Tieši šādu apsvērumu dēļ, dota posma veikšanai ir pielietota LINQ to SQL tehnoloģija.

3. DARBVIRSMAS LIETOJUMPROGRAMMAS IZSTRĀDE

Dotajā nodaļā tiek aprakstīta darbvirsmas lietojumprogramma, kura tika izstrādāta ar mērķi piedāvāt kursu reģistrēšanas iespējas un ļaut lietotājam saņemt pilnu informāciju par izvēlēto kursu.

3.1. Veiktās darbības

Kursu reģistrēšanas darbvirsmas lietojumprogramma tika izstrādāta ar .NET platformas grafiskas bibliotēkas *Windows Forms* palīdzību. Tā ietver sevī vairākas kontroles, kuras tiek grupētas atsevišķas *GroupBox* kontroles un ļauj gan pieteikt kursus, ievadot nepieciešamo informāciju, gan arī izvēlēties no saraksta vēlamo kursu. Kursu reģistrēšanas darbvirsmas lietojumprogramma ir attēlota 3.1.1 attēlā.

3.1.1 att. Kursu reģistrēšanas lietotāja forma.

Ievades lauku validācija tiek nodrošināta ar kontroles *ErrorProvider* palīdzību. Dotajā gadījumā jābūt aizpildītiem visiem teksta laukiem, maksimālam dalībnieku skaitam jābūt veselam pozitīvam skaitlim un kursu norises datumiem jābūt norādītiem nākotnē attiecībā pret pašreizējo sistēmas laiku, ka arī atbilstošiem attiecība viens pret otru. *ErrorProvider* kontroles pielietošanas piemērs izskatās sekojoši:

```
private void tbValue_Validating(object sender, CancelEventArgs e) {  
    TextBox tb = (TextBox)sender;  
    e.Cancel = true;  
  
    if (tb != null) {
```

```

        errorProvider.SetError(tb, null);

        if (tb.Text == String.Empty) {
            errorProvider.SetError(tb, "Laukam jābūt aizpildītam.");
            return;
        }

        e.Cancel = false;
    }
}

```

Augstāk attēlota pirmtekstā tiek validēti visi teksta lauki attiecībā pret ievadīto vērtību. Dotais notikums tiek izraisīts kursa reģistrēšanas laikā ar metodes *ValidateChildren* palīdzību. Kontroles validācijas rezultāts ir attēlots 3.1.2 attēlā.

3.1.2 att. Ievadlauku validācijas piemērs.

Maksimālo dalībnieku ievades lauks papildus tiek validēts ar *KeyPress* notikuma palīdzību (rodas simbola ievades laikā). Tiek ignorētas visas vērtības, kuras nav skaitļi vai arī *backspace* taustiņa kods. Pārbaudes pirmkods izskatās sekojoši:

```

private void txtMaxParticipants_KeyPress(object sender,
    KeyPressEventArgs e) {
    if (!char.IsDigit(e.KeyChar) &&
        e.KeyChar != (char)8)
        e.Handled = true;
}

```

3.2. Posma secinājumi

Posma izpilde nesagādāja nekādas īpašas problēmas. Pielietojot Windows formu komponentus, kontroļu īpašību un notikumu kopumu ir iespējams sastādīt profesionālo

lietotāja saskarni, paredzot nepieciešamo ievaddatu validāciju, notikumu izraisīšanu, funkcionalitātes implementēšanu u.tml. Lietotāja saskarnes izstrādāšana tiek liela mēra atvieglota ar Visual Studio grafiska dizainera palīdzību.

4. TĪMEKĻA VIETNES IZSTRĀDE

Dotajā nodaļā tiek apskatīta ASP.NET tīmekļa vietne, kuras mērķis ir piedāvāt lietotājiem pieteikšanas iespēju kādiem noteiktiem kursiem. Dota uzdevuma veikšanai, lietotājam ir jāizvēlas noteiktu kursu un jāievada savus personas datus.

4.1. Veiktās darbības

Tīmekļa vietne sastāv no divām ASP.NET tīmekļa formām. Pirmā forma veic pilnas informācijas izvadi par kursiem un piedāvā pieteikšanas iespēju (skat. 4.1.1 attēlu).

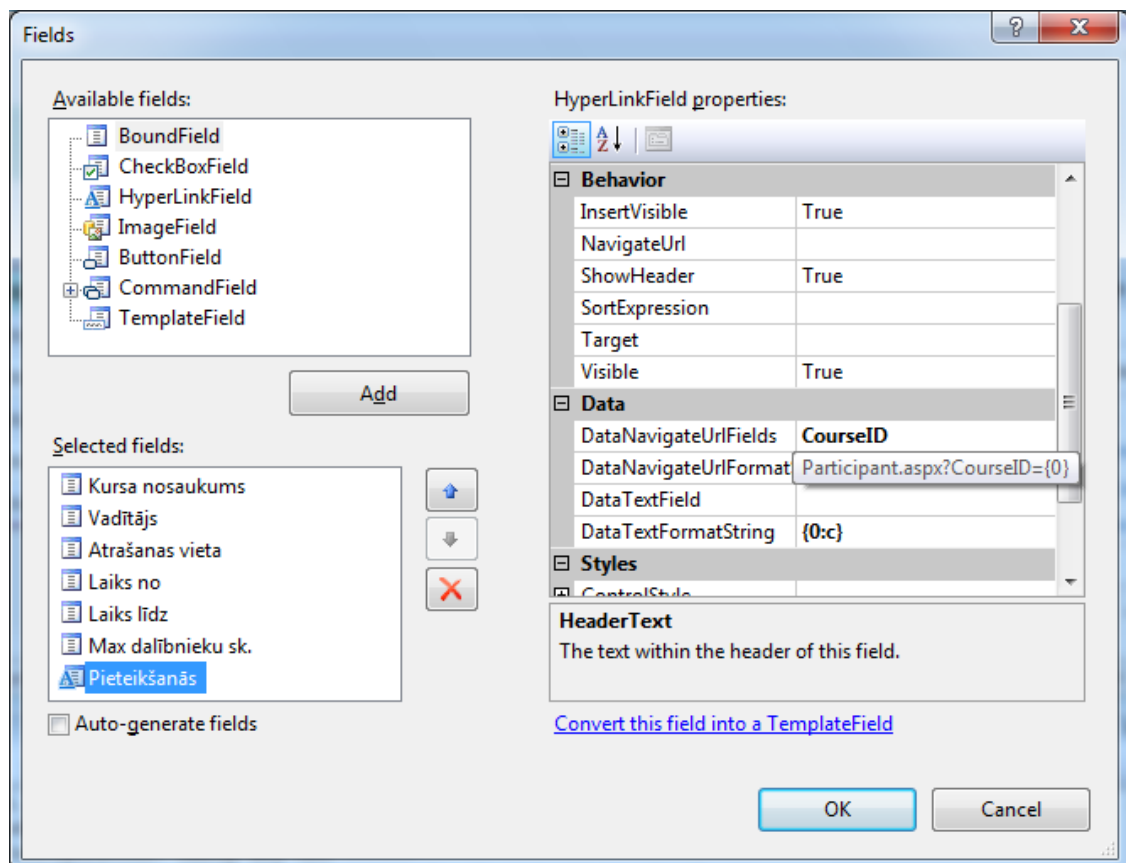
Reģistrēšanās kursiem						
Kursa nosaukums	Vadītājs	Atrašanās vieta	Laiks no	Laiks līdz	Max dalībnieku sk.	Pieteikšanās
Filozofija	Inguna Šablinska	Rīga, Maskavas 123, 123	29.09.2011 12:57	30.09.2011 12:57	122	Pieteikties
Alķīmija	Vasilijs Pupkins	Rīga, Tallinas 44, 217	06.10.2010 10:00	06.10.2010 17:00	55	Pieteikties
Filoloģija	Inga Makarova	Rīga, Mārijas 23, 222	01.10.2010 10:00	01.10.2010 14:00	15	Pieteikties
Politoloģija	Linda Medne	Rīga, Mārijas 23, 125	30.09.2010 10:00	30.09.2010 17:00	30	Pieteikties
Datorzinātnes	Ēriks Mihailovs	Rīga, Mārijas 23, 137	29.09.2010 09:00	29.09.2010 15:00	25	Pieteikties
Menedžments	Igors Grigorjevs	Rīga, Mārijas 23, 155	29.09.2010 09:00	29.09.2010 15:00	15	Pieteikties

4.1.1 att. Kursu pieteikšanas tīmekļa forma.

Kursu saraksta izvadei ir pielietota *GridView* kontrole. Kontroles lauki tika norādīti ar *Fields* veidnes palīdzību (skat. 4.1.2 attēlu). Tai ir paredzēti seši datu lauki (*BoundField*) un viens saites lauks (*HyperLinkField*). Datu lauku saistīšana ar ierakstu kolonu laikiem tiek specificēta ar veidnes *DataField* īpašību; tajā tiek norādīta ielādējama ieraksta lauku kolona, kuru reprezentēs izvēlēta kontroles lauku kolona. Savukārt, saites laukā tiek norādīta parametru virkne, kuru paredz izvēlēta kursa unikāla identifikatora nodošanu otrai ASP.NET tīmekļa formai, kas pilda kursa dalībnieka reģistrēšanu. Šādi līdz ar ievadītiem personas datiem tiek ieglabāts arī izvēlēta kursa unikāls identifikators.

GridView kontrole tiek aizpildīta formas ielādes laikā. Kontroles aizpildīšanas pirmteksts izskatās sekojoši:

```
protected void Page_Load(object sender, EventArgs e) {
    if (!IsPostBack) {
        try {
            gwCourses.DataSource = GetMethods.GetCourses();
            gwCourses.DataBind();
        }
        catch (Exception ex) {
            Debug.WriteLine("Error: " + ex.Message);
        }
    }
}
```



4.1.2 att. GridView kontroles lauku specificēšana.

Otrā tīmekļa formā ir izvietotas vairākas ASP.NET kontroles, kurās tiek veikta dalībnieka ievaddatu ievade (skat. 4.1.3 attēlu).

Reģistrēšanās kursiem

Vārds:

Uzvārds:

E-pasts:

Tālrunis:

28 + 15 =

Izteiksme ir aprēķināta nepareizi.

4.1.3 att. Dalībnieku pieteikšanas forma.

Formā ir paredzēts CAPTCHA lauks, kas veic gadījuma matemātiskas izteiksmes ģenerēšanu. Šāda funkcija tiek nodrošināta ar sekojošo pirmteksta palīdzību:

```
private void GenerateExpression() {
    op1 = random.Next(100);
    op2 = random.Next(100);

    action = (Operation)Enum.ToObject(typeof(Operation), random.Next(2));
    lblExp.Text = String.Format("{0} {1} {2}", op1, OpToStr(action), op2);
    result = CalculateExpression(op1, action, op2);
    ViewState["result"] = result;
}
```

Pēc pirmteksta var secināt, ka sākotnēji tiek ģenerēti divi gadījuma izteiksmes skaitļi, tad atbilstoša matemātiska operācija (tiek iegūta no *Operation* pārskaitījuma). Turpmāk, izsaucot *CalculateExpression* metodi, tiek iegūts izteiksmes aprēķinātais rezultāts, kas tiek saglabāts tīmekļa lapas *ViewState* stāvoklī.

Formu kontroļu vērtības tiek validētas ar *CustomValidator* kontroles palīdzību. Vērtību validācija notiek dalībnieka ievaddatu ieglabāšanas gaitā, izpildoties kontroles metodei *Validate*. Dotās darbības rezultātā tiek izpildīts zemāk atspoguļots validācijas pirmkoda fragments:

```
protected void registrationValidator_ServerValidate(object source,
    ServerValidateEventArgs args) {

    Regex emailRegex = new Regex("(?<user>[^@]+)@(?<host>.+");
    args.IsValid = true;

    if (String.IsNullOrEmpty(txtEmail.Text) ||
        txtEmail.Text.Length > 50 ||
        !emailRegex.Match(txtEmail.Text).Success) {

        args.IsValid = false;
        registrationValidator.Text = "E-pasts nav korekti ievadīts.";
        return;
    }

    ...

    result = (int)ViewState["result"];
    if (number != result) {
        args.IsValid = false;
        registrationValidator.Text = "Izteiksme ir aprēķināta nepareizi.";
        return;
    }
}
```

Kursa dalībnieka datu ieglabāšanas koda fragments izskatās sekojoši:

```

protected void btnSave_Click(object sender, EventArgs e) {
    registrationValidator.Validate();

    if (!registrationValidator.IsValid) return;

    registrationValidator.Text = String.Empty;
    int courseId = int.Parse(Request.QueryString["CourseID"]);

    try {
        if (GetMethods.IsParticipated(courseId,
            Server.HtmlDecode(txtFirstname.Text),
            Server.HtmlDecode(txtLastname.Text),
            int.Parse(Server.HtmlDecode(txtPhoneNum.Text)))) {

            msgLiteral.Text = "Jūs esat jau reģistrēts/-a dotajām kursam.";
            return;
        }

        msgLiteral.Text = String.Empty;
        Participant participant;

        participant = new Participant {
            Firstname = txtFirstname.Text,
            Lastname = txtLastname.Text,
            Email = txtEmail.Text,
            PhoneNumber = int.Parse(txtPhoneNum.Text),
            CourseID = courseId,
            RegistrationDate = DateTime.Now.Date
        };

        AddMethods.AddParticipant(participant);

        msgLiteral.Text = "Jūs esat vieksmīgi reģistrējies/-usies
            izvēlētajam kursām.";

        ClearControls();
        GenerateExpression();
    } catch (Exception ex) {
        Debug.WriteLine("Error: " + ex.Message);
    }
}

```

Izpildoties pirmtekstam, sākotnēji tiek veikta ievaddatu validācija. Turpmāk, seko dalībnieka atkārtotas ievades pārbaude, t.i., vai viņš ir tiek reģistrēts atkārtoti un tikai tad dalībnieka reģistrēšana sistēmā.

4.2. Posma secinājumi

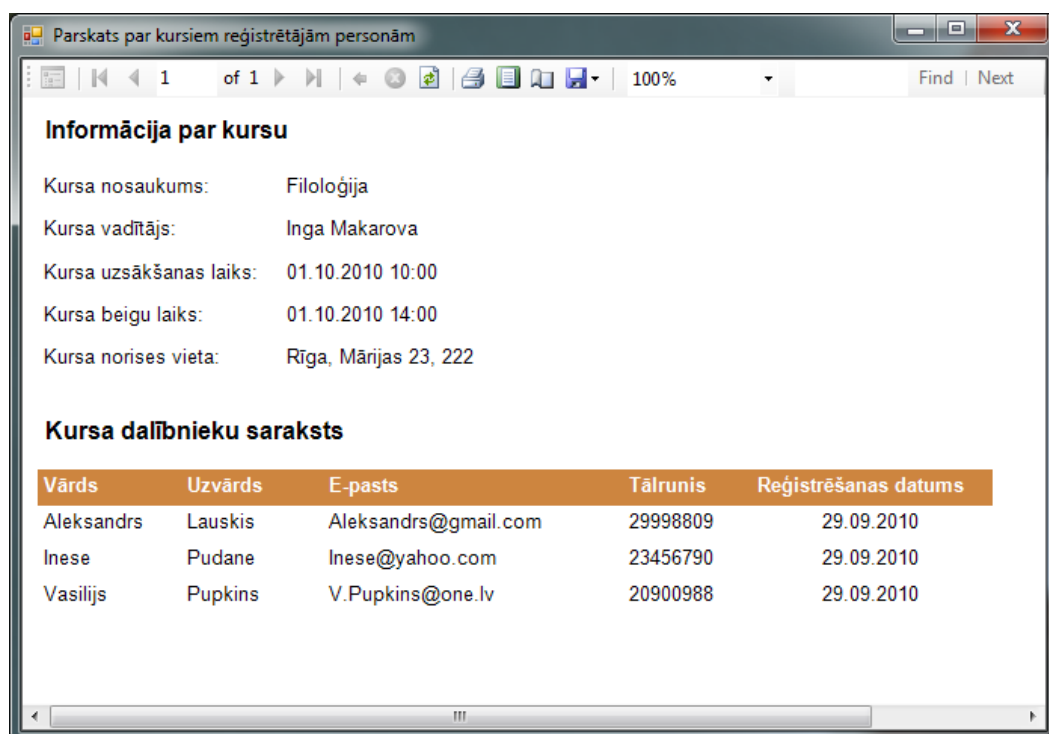
Izpildītais posms ļāva apgūt ASP.NET tehnoloģijas galvenās koncepcijas, kuras saistās ar tīmekļa vietnes dzīves ciklu. Posma ietvaros pielietotas kontroles atviegloja tīmekļa formu izstrādi un datu vadību tajās. Visual Web Developer redaktora pielietojums uzlaboja tīmekļa vietnes izstrādes procesu, piedāvājot interaktīvu kontroļu metožu un īpašību, ka arī CSS stilu atspoguļošanas mehānismu.

5. PĀRSKATA IZTRĀDE

Dotajā nodaļā tiek aprakstīts pārskata izstrādes posms, kura mērķis ir izvadīt pilno informāciju par kursu un tā dalībniekiem.

5.1. Veiktās darbības

Posma izpildes gaitā izstrādātais pārskats ir attēlots 5.1.1 attēlā. Par pamattehnoloģiju pārskata izstrādei ir izvēlēta Microsoft Reporting Services tehnoloģija.



Informācija par kursu

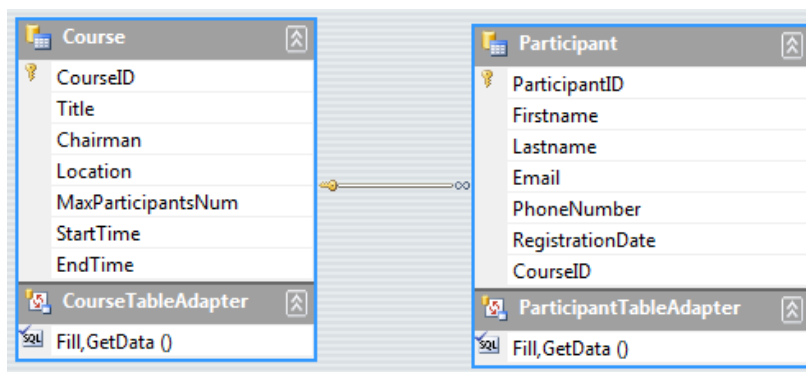
Kursa nosaukums: Filoloģija
Kursa vadītājs: Inga Makarova
Kursa uzsākšanas laiks: 01.10.2010 10:00
Kursa beigu laiks: 01.10.2010 14:00
Kursa norises vieta: Rīga, Mārijas 23, 222

Kursa dalībnieku saraksts

Vārds	Uzvārds	E-pasts	Tālrunis	Reģistrēšanas datums
Aleksandrs	Lauskis	Aleksandrs@gmail.com	29998809	29.09.2010
Inese	Pudane	Inese@yahoo.com	23456790	29.09.2010
Vasilijs	Pupkins	V.Pupkins@one.lv	20900988	29.09.2010

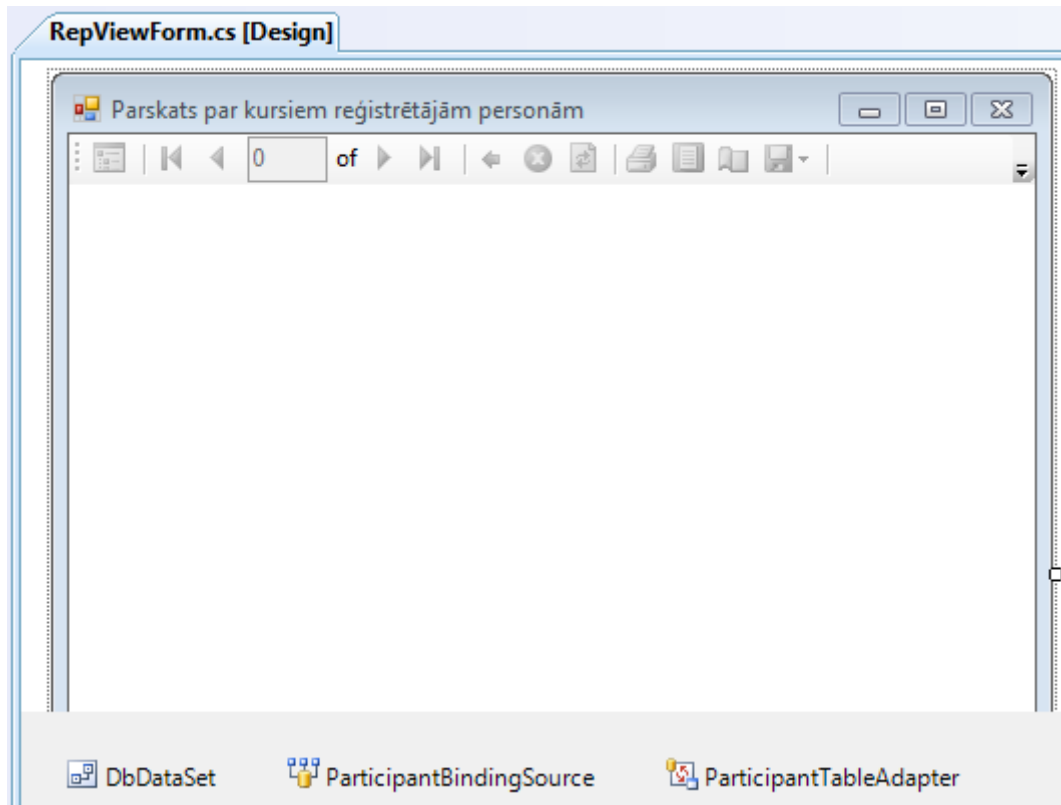
5.1.1 att. Izstrādātais pārskats.

Pārskata izstrādei sākotnēji ir paredzēts *DataSet* objekts, kurš veic nepieciešamo datu izgūšanu gan no kursu, gan no dalībnieku tabulām (skat. 5.1.2 attēlu).



5.1.2 att. *DataSet* objekta shēma.

Turpmāk ir paredzēta pārskata ģenerēšanas forma, kura tiek izsaukta veicot nepieciešama kursa izvēli no iepriekš apskatītas formas (skat. 3.1.1 attēlu). Tajā ir izvietota *MicrosoftReportViewer* kontrole, kas nodrošina konstruēta pārskata attēlošanu un vairākas nevizuālas komponentes datu ielādēšanai no datu bāzes (skat. 5.1.3).



5.1.3 att. Pārskata izvades forma.

Nākamā uzdevuma izpildes posmā līdz ar grafiska redaktora palīdzību ir konstruēti vairāki lauki, kas ir paredzēti pārskatām nododamo parametru attēlošanai, ka arī tiek izvietota kursam piederošo dalībnieku datu izvades tabula. Dotie dati tiek atlasīti atbilstoši pieprasītām kursam un kārtoti pēc dalībnieku uzvārdiem augoša secībā. Konstruējamais pārskats ir attēlots 5.1.4 attēlā.

Pēdējā uzdevuma izpildes posmā pārskata ģenerēšanai un tām nepieciešamo parametru nodošanai tiek izstrādāts sekojošs pirmteksts:

```
private void btnGenerateReport_Click(object sender, EventArgs e) {
    Course course = GetMethods.
        GetCourseByID((int)lstCourses.SelectedValue);
    RepViewForm reportForm = new RepViewForm();
    var reportParams = new ReportParameter[6];

    reportParams[0] =
        new ReportParameter("courseId", course.CourseID.ToString());
    reportParams[1] = new ReportParameter("courseTitle", course.Title);
}
```

```

reportParams[2] =
    new ReportParameter("courseChairman", course.Chairman);
reportParams[3] =
    new ReportParameter("courseLocation", course.Location);
reportParams[4] = new ReportParameter("courseStartTime",
    course.StartTime.ToString("dd/MM/yyyy HH:mm"));
reportParams[5] = new ReportParameter("courseEndTime",
    course.EndTime.ToString("dd/MM/yyyy HH:mm"));

reportForm.reportViewer.LocalReport.SetParameters(reportParams);
reportForm.ShowDialog();
}

```

The screenshot shows the Microsoft Access Report Designer interface. The top tab is 'CoursePartRep.rdlc [Design]'. The report is divided into two main sections: 'Page Header' and 'Body'. The 'Page Header' section contains a title 'Informācija par kursu' and five text boxes with labels and parameter references. The 'Body' section contains a table titled 'Kursa dalībnieku saraksts' with five columns: Vārds, Uzvārds, E-pasts, Tālrunis, and Reģistrēšanas datums. The table rows show data from the 'Fields!' collection.

Vārds	Uzvārds	E-pasts	Tālrunis	Reģistrēšanas datums
=Fields!Firstna	=Fields!Lastnar	=Fields!Email.Value	=Fields!Phone	=Format(Fields!Registration

5.1.4 att. Pārskata konstruēšana redaktorā.

5.2. Posma secinājumi

Veicot posma izstrādi ir iegūtas iemaņas darbā ar pārskatiem. Pārskata izstrāde prasīja papildus laiku kursa un tā dalībnieku datu apvienošanas iespējas realizēšanai. Grafiska redaktora pielietojums nerādīja īpašas grūtības, jo bija iegūtas iemaņas darbā ar līdzīgo redaktoru formu izstrādei MS Access lietojumprogrammā.

NOBEIGUMS

Sistēmas izstrādes gaitā tika paplašinātas iemaņas darbā ar Windows Forms, ASP.NET un pārskatu veidojošiem komponentiem. Darbs ļāva pielietot esošas zināšanas un iegūt jaunas.

Esošo zināšanu pielietošana izpaudās posmos, kuros tika pielietotas LINQ to SQL, Windows Forms un ASP.NET tehnoloģijas. Posmu izpildes gaitā izstrādātie risinājumi bija balstīti uz iepriekš iegūto pieredzi līdzīgo uzdevumu veikšanā. Kaut gan veicot uzdevumus, tika mēģināts atrast optimālākos risinājumus, ka rezultātā tika paplašinātas esošas iemaņas un iegūtas jaunas zināšanas.

Jauno zināšanu iegūšana galvenokārt saistījās ar pārskata izstrādi. Dota uzdevuma izpildes rezultātā tika iepazīta *MicrosoftReportViewer* kontrole un iegūtas iemaņas darbā ar pārskatu konstruēšanas redaktoru.

Uzdevuma izpildes gaitā tika uzlabotas iemaņas darbā ar Visual Studio izstrādes vidi.

LITERATŪRA

- [Metha 2008] Mehta P. V. Pro LINQ Object Relational Mapping with C# 2008 – New York: Springer-Verlag, 2008. – 383. lpp.