



Teamprojekt II

Dokumentation der XML Manipulation

Name, Vorname	Roshanzadeh, Amirhossein
Matrikelnummer	19660
Name, Vorname	Barat Khooei, Ahmad
Matrikelnummer	20164
Semester	Sommersemester 2023
Fachbereich	Fakultät für Elektrotechnik und Informatik
Modul	INFM2300 Teamprojekt II
Eingereicht bei	Prof. Dr. sc. hum. Wilfried Honekamp
Abgabedatum	30.08.2023

Einleitung.....	1
XML-Manipulation.....	2
Warum XML-Manipulation?.....	2
Grundlegende Operationen der XML-Manipulation:.....	2
Lesen von XML-Daten:.....	2
Schreiben von XML-Daten:.....	2
Aktualisieren von XML-Daten:.....	2
Löschen von XML-Daten:.....	2
XML-Manipulation in C# mit Windows Forms:.....	3
Codebeispiele.....	3
Browse Button.....	3
ShowDetails Button.....	4
EditSelectedRow Button.....	4
Edit Button.....	5
EditVorankmeldungen.....	5
EditPlatz Button.....	6
Export XML Button.....	7
Arbeiten mit der Benutzeroberfläche.....	8
Browser-Button.....	9
Spieler Rankings Form.....	9
ShowDetails Button.....	10
Add(ShowDetails).....	10
Edit (ShowDetails).....	11
Delete (ShowDetails).....	12
Save (ShowDetails).....	12
Exit (ShowDetails).....	12
Edit Button.....	13
Export XML Button.....	13
Edit(Spiel_Ranking_Form).....	14
Mein Beitrag.....	14
Amirhossein Roshanzadeh:.....	14
Ahmad Barat Khooei:.....	14
Quellen.....	16

Einleitung

Diese Ausarbeitung wird im Rahmen des Moduls INFM2300 Teamprojekt II erstellt. Das Ziel ist es, in einer Gruppe von zwei Studierenden eine Applikation für eine XML Manipulation zu erarbeiten. Die zwei an dieser Arbeit beteiligten Studenten sind Amirhossein Roshanzadeh(19660), Ahmad Barat Khooei(20164) .

in diesem Dokument werden wir Ihnen einen umfassenden Überblick über die Funktionalitäten, die Verwendung und die technischen Aspekte meiner Anwendung geben. Diese Anwendung wurde entwickelt, um die effiziente Manipulation von XML-Daten zu ermöglichen und gleichzeitig die Integrität und Struktur der XML-Dateien zu erhalten.

"Unser Projekt verfolgt zwei Hauptziele: Zum einen ermöglicht es die Manipulation von XML-Daten durch Funktionen wie Bearbeiten, Hinzufügen, Speichern und Löschen. Zum anderen bietet es die Möglichkeit, Spieler-Rankings auszutauschen und auszutauschen."

XML-Manipulation

XML (Extensible Markup Language) ist eine weit verbreitete Auszeichnungssprache, die zur Darstellung strukturierter Daten verwendet wird. XML-Dokumente bestehen aus hierarchisch organisierten Elementen, die Informationen in einem les- und maschinen verarbeitbaren Format speichern. XML-Manipulation bezieht sich auf die Bearbeitung und Modifikation von XML-Dokumenten, sei es das Hinzufügen neuer Daten, das Ändern von bestehenden Informationen oder das Entfernen von Elementen.

Warum XML-Manipulation?

Die Notwendigkeit zur XML-Manipulation ergibt sich aus der Flexibilität und Verbreitung von XML-Dateiformaten. Anwendungen müssen oft XML-Daten generieren, analysieren oder transformieren, um Informationen zwischen Systemen auszutauschen oder sie in einer benutzerfreundlichen Darstellung anzuzeigen.

Grundlegende Operationen der XML-Manipulation:

Lesen von XML-Daten:

XML-Dokumente können gelesen werden, um Informationen auszulesen. Dies beinhaltet das Navigieren durch die hierarchische Struktur und das Extrahieren von Elementen und Attributen.

Schreiben von XML-Daten:

Neue XML-Daten können erstellt oder bestehende geändert werden, um aktualisierte Informationen zu speichern. Dies beinhaltet das Erstellen von Elementen, das Festlegen von Attributwerten und das Strukturieren des Dokuments.

Aktualisieren von XML-Daten:

Bestehende XML-Daten können bearbeitet werden, um Informationen zu aktualisieren oder zu korrigieren. Dies beinhaltet das Ändern von Elementinhalt und Attributen.

Löschen von XML-Daten:

Elemente oder sogar ganze Zweige eines XML-Dokuments können entfernt werden, um nicht mehr benötigte Informationen zu bereinigen.

XML-Manipulation in C# mit Windows Forms:

Die Entwicklungsumgebung C# bietet eine mächtige Plattform für die XML-Manipulation. Mit der Integration von Windows Forms können Benutzeroberflächen erstellt werden, um XML-Dateien auszuwählen, zu bearbeiten und zu speichern. Windows Forms ermöglichen eine benutzerfreundliche Interaktion mit der Anwendung, während C#-Code die XML-Manipulation hinter den Kulissen durchführt

Codebeispiele

Browse Button

```
private void btnBrowse_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    // Set the initial directory and filter for the file dialog
    openFileDialog.InitialDirectory = "C:/Users/User/Desktop";
    openFileDialog.Filter = "XML Files (*.xml)|*.xml";

    // Show the file dialog and check if the user clicked the OK button
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog.FileName;

        // Load the XML file using the selected file path
        xmlDoc.Load(filePath);
        loadXml();

        // Process the XML file as needed
        // ...
    }
}
```

ermöglicht der Code dem Benutzer, eine XML-Datei auszuwählen, lädt diese XML-Datei in ein XML-Dokument-Objekt und ruft dann die Methode loadXml() auf, um mit der verarbeiteten XML-Datei weiterzuarbeiten.

ShowDetails Button

```
private void btnShowDetails_Click(object sender, EventArgs e)
{
    EditForm editForm = new EditForm(xmlDoc);
    editForm.ShowDialog();
    editForm.FormClosed += EditForm_FormClosed;
}

private void EditForm_FormClosed(object sender, FormClosedEventArgs e)
{
    EditForm editForm = (EditForm)sender;
    ReturnedObject = editForm.ReturnedObject;
}
```

Dieser Code behandelt das Bearbeiten und Löschen von Datensätzen in einer DataGridView und aktualisiert die zugrunde liegende Datenquelle entsprechend. Es scheint auch XML-Operationen für das Hinzufügen und Entfernen von Elementen auszuführen, jedoch fehlt der Codekontext, um die genaue Funktionalität zu verstehen.

EditSelectedRow Button

```
13 public partial class EditSelectedRow : Form
14 {
15     public DataGridViewRow selectedRow;
16     private bool shouldHandleTextChanged = true;
17
18     public EditSelectedRow(DataGridViewRow selectedRow)
19     {
20         InitializeComponent();
21         this.selectedRow = selectedRow;
22         loadXml();
23     }
24
25     private void loadXml()
26     {
27         txtPos.Text = selectedRow.Cells["Pos"].Value.ToString();
28         string[] FullName = selectedRow.Cells["Team"].Value.ToString().Split(' ');
29         txtFirstName.Text = FullName[0];
30         txtLastName.Text = FullName[1];
31         txtDisziplin.Text = selectedRow.Cells["Disziplin"].Value.ToString();
32     }
33
34     private void EditSelectedRow_Load(object sender, EventArgs e)
35     {
36     }
37
38     private void btnExit_Click(object sender, EventArgs e)
39     {
40         Close();
41     }
42
43     private void btnSave_Click(object sender, EventArgs e)
44     {
45         selectedRow.Cells["Pos"].Value = txtPos.Text;
46         selectedRow.Cells["Team"].Value = txtFirstName.Text + " " + txtLastName.Text;
47         selectedRow.Cells["Disziplin"].Value = txtDisziplin.Text;
48         EditForm originalForm = (EditForm)Application.OpenForms["EditForm"];
49         originalForm.UpdateDataGridViewRow(selectedRow);
50         Close();
51     }
52
53     private void txtDisziplin_TextChanged(object sender, EventArgs e)
54     {
55     }
56
57
58
59     private void txtDisziplin_KeyPress(object sender, KeyPressEventArgs e)
60     {
61         lblDisziplinChanged.Text = "All Disziplin Will be Changed";
62     }
63
64
65
66 }
```

Der Code insgesamt implementiert also ein Bearbeitungsformular für eine ausgewählte Zeile in einer DataGridView. Es ermöglicht dem Benutzer, Änderungen in den Werten der Zeile vorzunehmen und speichert diese Änderungen dann zurück in der DataGridView und im Hauptformular.

Edit Button

```
private void btnEditRow_Click(object sender, EventArgs e)
{
    // Get the selected row
    DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];
    rowIndex4Change = selectedRow.Index;
    if (dataGridView1.SelectedRows.Count == 1)
    {
        if (displayPlatzTable)
        {
            new EditPlatz(selectedRow).ShowDialog();
            sortDataGridView();
        }
        else
        {
            new EditVorankmeldungen(selectedRow).ShowDialog();
            // Access the values of the selected row
        }
    }
    else
    {
        MessageBox.Show("Please Select just one Row..");
    }
}
```

ermöglicht der Code dem Benutzer, eine ausgewählte Zeile in einer DataGridView zu bearbeiten. Je nach Wert der Display- PlatzTable-Variable wird entweder das Edit-Platz-Formular oder das Edit-Vorankmeldungen-Formular geöffnet. Nach dem Bearbeiten wird die DataGridView gegebenenfalls neu sortiert.

EditVorankmeldungen

```
public partial class EditVorankmeldungen : Form
{
    public DataGridViewRow selectedRow;

    public EditVorankmeldungen(DataGridViewRow selectedRow)
    {
        InitializeComponent();
        this.selectedRow = selectedRow;
        loadXml();
    }

    private void loadXml()
    {
        txtDatum.Text = selectedRow.Cells["Datum"].Value.ToString();
        txtTurnier.Text = selectedRow.Cells["Turnier"].Value.ToString();
        txtOrt.Text = selectedRow.Cells["Ort"].Value.ToString();
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        selectedRow.Cells["Datum"].Value = txtDatum.Text;
        selectedRow.Cells["Turnier"].Value = txtTurnier.Text;
        selectedRow.Cells["Ort"].Value = txtOrt.Text;
        FrmMain originalForm = (FrmMain)Application.OpenForms["FrmMain"];
        originalForm.UpdateDataGridView(selectedRow);
        Close();
    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        Close();
    }
}
```

Der Code implementiert also ein Formular, das es dem Benutzer ermöglicht, Voranmeldungsdaten zu bearbeiten. Die Methoden und Ereignisse des Formulars ermöglichen das Laden der Daten aus der ausgewählten Zeile, das Speichern von Änderungen in der DataGridView und das Schließen des Formulars.

EditPlatz Button

```
13 public partial class EditPlatz : Form
14 {
15     public DataGridViewRow selectedRow;
16     public EditPlatz(DataGridViewRow selectedRow)
17     {
18         InitializeComponent();
19         this.selectedRow = selectedRow;
20         loadData();
21     }
22
23     private void loadData()
24     {
25         txtPlatz.Text = selectedRow.Cells["Platz"].Value.ToString();
26         txtPlayer.Text = selectedRow.Cells["player"].Value.ToString();
27     }
28
29     private void btnExit_Click(object sender, EventArgs e)
30     {
31         Close();
32     }
33
34     private void btnSave_Click(object sender, EventArgs e)
35     {
36         selectedRow.Cells["Platz"].Value = txtPlatz.Text;
37         selectedRow.Cells["player"].Value = txtPlayer.Text;
38         FrmMain originalForm = (FrmMain)Application.OpenForms["FrmMain"];
39         originalForm.UpdateDataGridViewRow(selectedRow);
40         Close();
41     }
42 }
43
44
```

erlaubt dieser Code das Bearbeiten von Platzierungs Daten in einem separaten Formular ("EditPlatz"). Die Methoden und Ereignisse des Formulars ermöglichen das Laden der Daten aus der ausgewählten Zeile, das Speichern von Änderungen in der DataGridView und das Schließen des Formulars.

Export XML Button

```
private void btnExportXML_Click(object sender, EventArgs e)
{
    if (ReturnedObject == null)
    {
        ReturnedObject = xmlDoc;
    }

    if (!displayPlatzTable)
    {
        exportVor anmeldungentXML();
    }
    else
    {
        exportPlatztXML();
    }

    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "XML files (*.xml)|*.xml";
    saveFileDialog.Title = "Save XML File";
    saveFileDialog.ShowDialog();

    if (saveFileDialog.FileName != "")
    {
        string filePath = saveFileDialog.FileName;
        ReturnedObject.Save(filePath);
        MessageBox.Show("The file has been saved successfully.");
    }
}
```

```
private void exportVor anmeldungentXML()
{
    #region Export Voranmeldungen XML

    string Turnierstart = string.Empty;
    string Name = string.Empty;
    string Ort = string.Empty;
    DataTable dt = dataGridView1.DataSource as DataTable;
    if (dt != null)
    {
        foreach (DataRow row in dt.Rows)
        {
            Turnierstart = row["Datum"].ToString();
            Name = row["Turnier"].ToString();
            Ort = row["Ort"].ToString();
        }
    }

    XmlNode voranmeldungenNode = ReturnedObject.SelectSingleNode("/root/Voranmeldungen");

    if (voranmeldungenNode != null)
    {
        voranmeldungenNode.Attributes["Turnierstart"].Value = Turnierstart;
        voranmeldungenNode.Attributes["Name"].Value = Name;
        voranmeldungenNode.Attributes["Ort"].Value = Ort;
    }

    #endregion
}
```

```
private void exportPlatztXML()
{
    List<string> platz = new List<string>();
    List<string> player = new List<string>();

    DataTable dt = dataGridView1.DataSource as DataTable;
    if (dt != null)
    {
        int index = 0;
        for (index = 0; index < dt.Rows.Count; index++)
        {
            DataRow row = dt.Rows[index];
            platz.Add(row["platz"].ToString());
            player.Add(row["Player"].ToString());

            XmlNode meldungNode = xmlDoc.SelectSingleNode(
                $"//meldung[@name='{player[index]}']");
            if (meldungNode != null)
            {
                meldungNode.Attributes["platz"].Value = platz[index];
                // teamNode.ParentNode.ParentNode.Attributes["BezeichnungLang"].Value = disziplin;
            }
        }
    }
}
```

Insgesamt ermöglicht der Code also das Exportieren von Daten aus einer DataGridView in eine XML-Datei. Die Methoden `exportVoranmeldungentXML` und `ExportPlatz XML` sind verantwortlich für das Zusammenstellen und Speichern der Daten im XML-Format. Der Button-Click-Handler organisiert den gesamten Exportprozess.

Arbeiten mit der Benutzeroberfläche

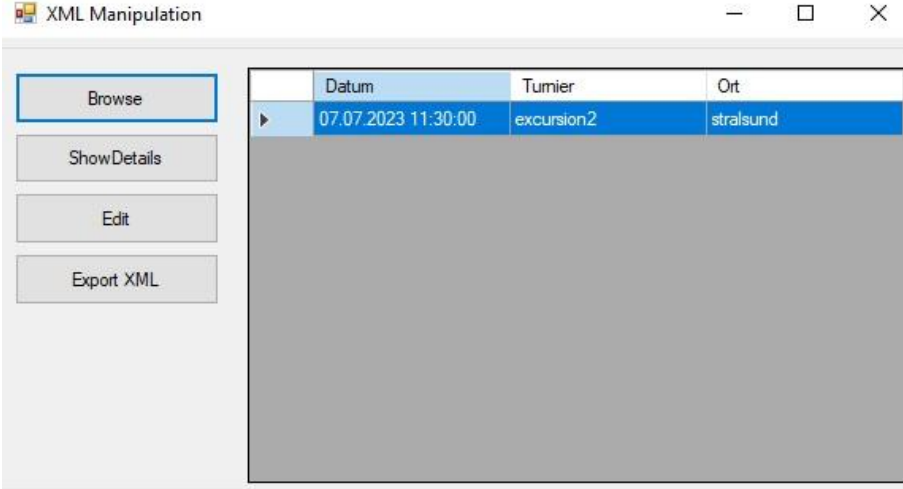


Auf der Hauptseite sind zu Beginn vier Button sichtbar, nämlich Browse, ShowDetails, Edit und Export XML

Browser-Button

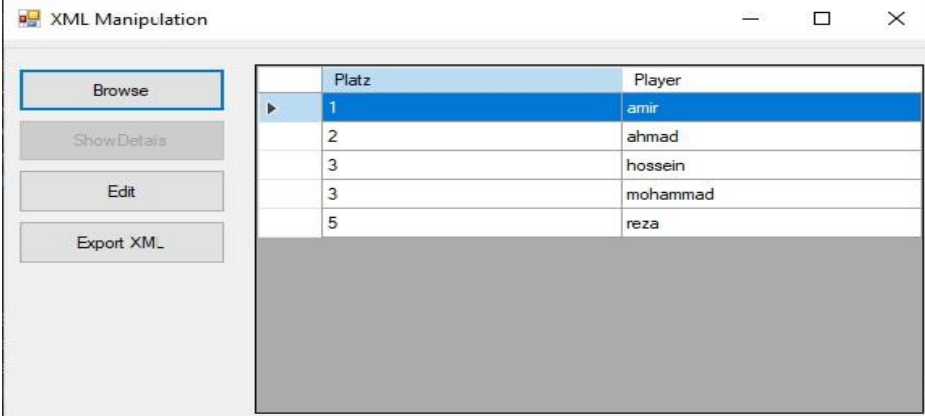
Der erste Button auf der Hauptseite ermöglicht es den Nutzern, den Filepath zu öffnen, um bequem nach gewünschten XML-Dateien zu suchen und zu importieren.

Mit dem Browser-Button steht Ihnen eine leistungsstarke Funktion zur Verfügung. Hiermit können Sie sowohl XML-Daten von Kicker Tools als auch XML-Daten von Tifu Tools importieren. Die Applikation ist in der Lage, diese Datenformate eigenständig zu interpretieren und zu verarbeiten



	Datum	Turnier	Ort
▶	07.07.2023 11:30:00	excursion2	stralsund

Voranmeldung Form



	Platz	Player
▶	1	amir
	2	ahmad
	3	hossein
	3	mohammad
	5	reza

Spieler Rankings Form

ShowDetails Button

Der zweite Button ermöglicht es den Anwendern, detaillierte Informationen zu einem ausgewählten Element anzuzeigen. Dies hilft, ein umfassenderes Verständnis zu gewinnen.

Hier sind vier Button **Add, Edit, Delete, Save und Exit** sichtbar

The screenshot shows a window titled "EditForm" with a sidebar on the left containing five buttons: "Add", "Edit", "Delete", "Save", and "Exit". The main area displays a table titled "Vorankmeldungen". The table has three columns: "Pos", "Team", and "Disziplin". The first row is selected, showing "1" in the "Pos" column, "amir roshan" in the "Team" column, and an empty "Disziplin" column.

Pos	Team	Disziplin
1	amir roshan	
2	Hamzeh Abbasi	
3	Badre Abdeslam	
4		
5	Ali Abo-Zaineab	

Add(ShowDetails)

The screenshot shows the "EditForm" window with the "Edit" button highlighted. A dialog box titled "EditSelectedRow" is open, displaying details for the selected row (Pos: 1). The dialog has fields for "Pos:", "FirstName:", "LastName:", and "Disziplin:". The "Disziplin:" field is a list box with several options, including "Men Singles" which is selected. The dialog also has "Save" and "Exit" buttons.

Pos	Team	Disziplin
1	Jan Graf	Men Singles
2	Sinan Ahmeti	Men Singles
3	Mehdi Abassi	Men Singles
4	Marco Abreu	Men Singles
5	Sascha Achhammer	Men Singles

EditSelectedRow

Pos:

FirstName:

LastName:

Disziplin:

- ☐ Open Doubles
- ☒ Open Singles
- ☐ Men Doubles
- ☒ Men Singles
- ☐ Women Doubles

Save Exit

Die Schaltfläche "Add" ermöglicht es dem Nutzer, neue Informationen oder Einstellungen hinzuzufügen. Wenn Sie auf die "Add" Schaltfläche klicken, öffnet sich ein neues Formular mit den Feldern "Pos" (Position), "FirstName" (Vorname), "LastName" (Nachname) und "Disziplin" (Disziplin), in denen Sie die erforderlichen Informationen eingeben können.

Edit (ShowDetails)

The screenshot shows a software application titled 'EditForm'. On the left, there is a vertical sidebar with five buttons: 'Add', 'Edit', 'Delete', 'Save', and 'Exit'. The 'Edit' button is highlighted with a blue border. The main area of the application displays a table with four columns: 'Pos', 'Team', and 'Disziplin'. The table has five rows of data. The first row is selected, and a dialog box titled 'EditSelectedRow' is open in front of it. This dialog box contains input fields for 'Pos:', 'FirstName:', and 'LastName:', and a list box for 'Disziplin:'. The 'Pos' field contains the value '1', 'FirstName' contains 'Jan', and 'LastName' contains 'Graf'. The 'Disziplin' list box shows several options with checkboxes: 'Open Doubles', 'Open Singles', 'Men Doubles', 'Men Singles', and 'Women Doubles'. The 'Men Singles' option is selected and highlighted in blue. At the bottom of the dialog box, there are 'Save' and 'Exit' buttons.

Pos	Team	Disziplin
1	Jan Graf	Men Singles
2	Sinan Ahmeti	Men Singles
3	Mehdi Abassi	Men Singles
4	Marco Abreu	Men Singles
5	Sascha Achhammer	Men Singles

EditSelectedRow

Pos: 1

FirstName: Jan

LastName: Graf

Disziplin:

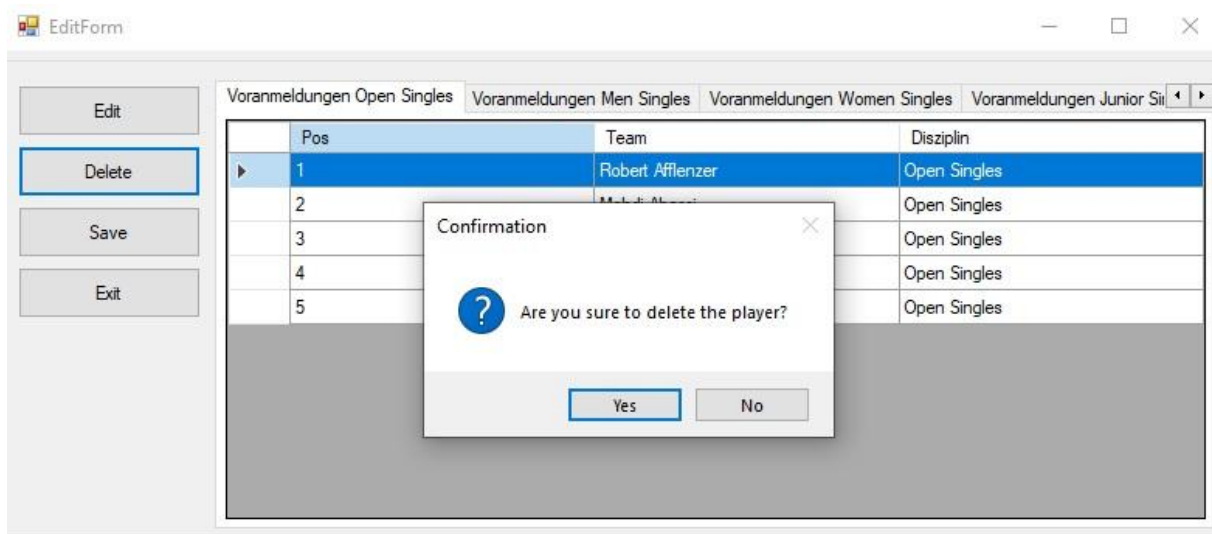
- ☐ Open Doubles
- ☒ Open Singles
- ☒ Men Doubles
- ☒ Men Singles
- ☐ Women Doubles

Save Exit

Diese Schaltfläche ermöglicht es dem Nutzer, den aktuellen Inhalt oder die Einstellungen zu bearbeiten. Durch das Anklicken der Edit Button öffnet sich ein Bereich, in dem Sie Änderungen vornehmen können.

In diesem Kontext öffnet sich ein neues Formular mit den Buttons Pos, FirstName, LastName, Disziplin

Delete (ShowDetails)



Durch Betätigen dieser Button können Sie ausgewählte Inhalte oder Elemente entfernen. Hiermit können unerwünschte oder veraltete Daten gelöscht werden.

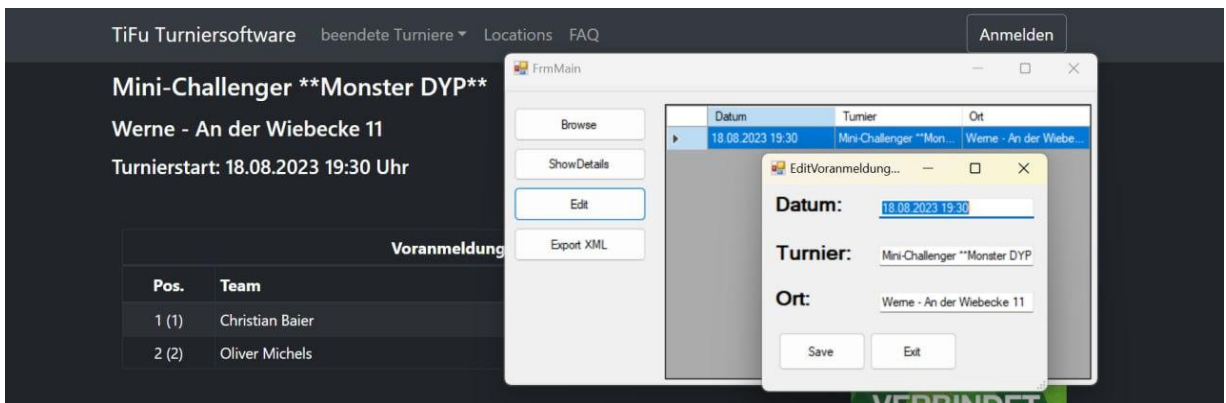
Save (ShowDetails)

Die "Save"-Schaltfläche dient dazu, alle vorgenommenen Änderungen oder Bearbeitungen zu speichern. Wenn Sie Anpassungen an Inhalten oder Einstellungen vorgenommen haben, müssen Sie auf diese Schaltfläche klicken, um die Änderungen zu sichern.

Exit (ShowDetails)

Mit dieser Schaltfläche können Sie die Applikation ordnungsgemäß beenden. Es schließt die Anwendung und beendet die aktuelle Sitzung. Es ist wichtig, darauf zu achten, dass alle relevanten Daten gespeichert wurden, bevor Sie die Applikation verlassen.

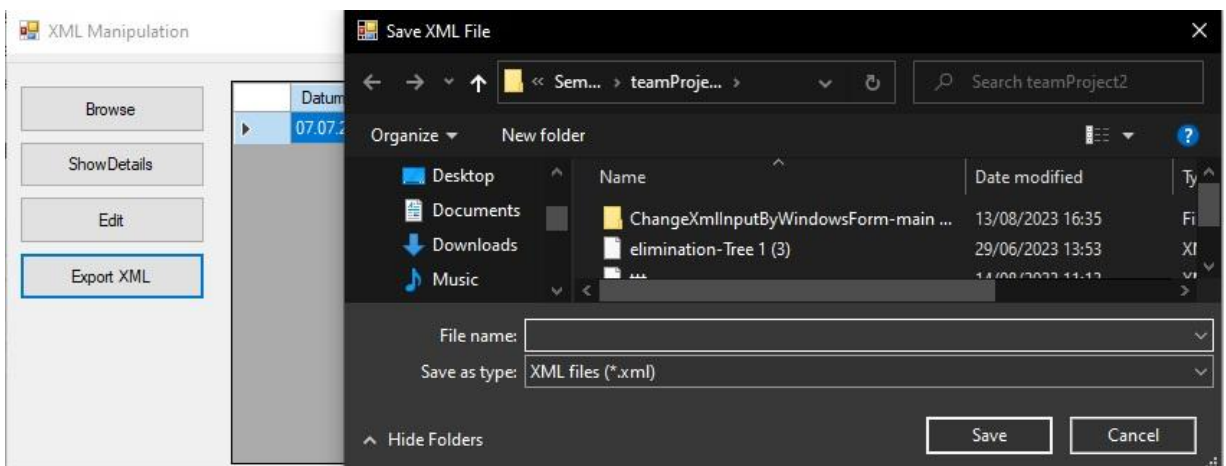
Edit Button



Mit dem dritten Button können Nutzer Inhalte bearbeiten und anpassen. Diese Funktion erlaubt es, Informationen individuell anzupassen und zu aktualisieren.

In diesem Kontext öffnet sich ein neues Formular mit den Buttons **Datum**, **Ort**, **Turnier**

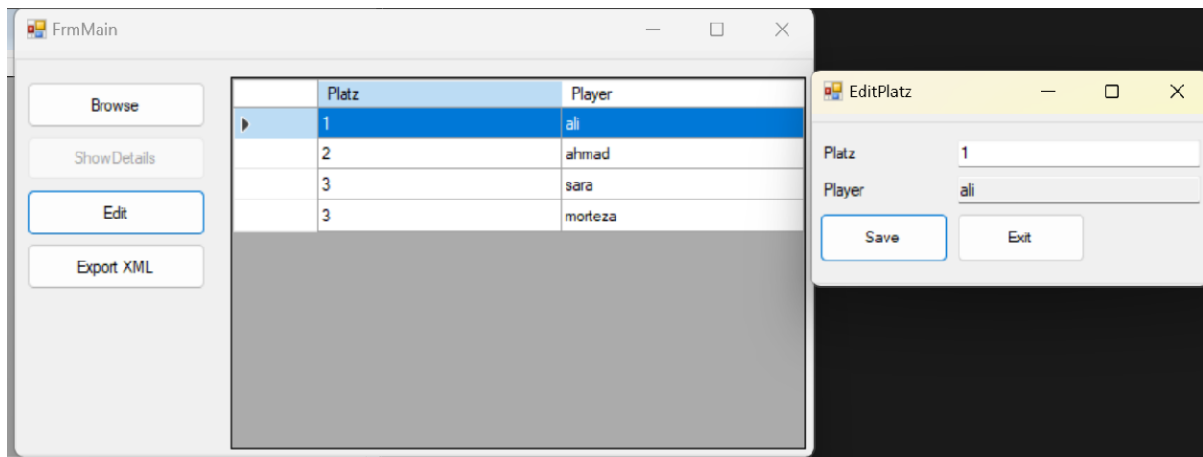
Export XML Button



Der vierte Button bietet die Möglichkeit, Daten in XML-Format zu exportieren.

Selbstständig erkennt dieser Button, welche Daten aus einer XML-Datei von Kicker Tools stammen und welche aus einer XML-Datei von Tifu Tools stammen.

Edit(Spiel_Ranking_Form)



In dieser Anwendung besteht die Option, die Platzierung von Spielern zu bearbeiten und auszutauschen. Ein neues Formular präsentiert den Platz des Spielers und den Namen des Spielers. "Nutzer haben die Befugnis, die Spieler Platzierungen zu manipulieren und auszutauschen, um eine individuelle Gestaltung vorzunehmen."

Mein Beitrag

Amirhossein Roshanzadeh:

Unter Verwendung von C# (Windows Form) habe ich im ersten Übungsteil das erste Hauptformular(Start Menü) erstellt und die Button "Edit" im Abschnitt 'ShowDetails' (dynamisch) gestaltet. Ich habe die entsprechenden Methoden und Codes geschrieben.

Sowohl in der ersten Übung als auch in der zweiten Übung habe ich die Schaltfläche "Export XML" entworfen und alle zugehörigen Methoden geschrieben.

In unserer Dokumentation habe ich von der Einleitung bis zum Abschnitt "Export Button" die Inhalte verfasst.

Ahmad Barat Khooei:

Mit Hilfe von C# (Windows Form) habe ich im ersten Übungsteil im Abschnitt "Edit" die Button "Edit selected Row" und "Edit Voranmeldung" erstellt. Im zweiten Übungsteil habe ich die Button "Save", "Delete", "Add" und Edit entworfen und die entsprechenden Methoden geschrieben.

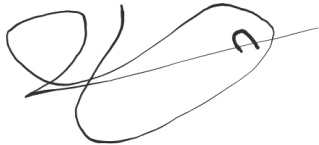
Die Dokumentation habe ich vom Abschnitt "Arbeiten mit der Software" verfasst.

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich:

1. dass ich diese Hausarbeit selbstständig verfasst habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,
3. dass ich diese Hausarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

A handwritten signature in black ink, consisting of a large, stylized loop followed by a horizontal stroke and a small 'n' at the end.

Amirhossein Roshanzadeh

A handwritten signature in black ink, featuring a series of connected loops and a long horizontal stroke at the end.

Ahmad Barat Khooei

Quellen

<https://darsman.com/courses/windows-foundation-course-csharp/>

<https://www.w3-farsi.com/posts/514/windows-form-in-csharp/>

<https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022>

<https://www.guru99.com/c-sharp-windows-forms-application.html>

<https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/>

<https://www.das-grosse-computer-abc.de/CSharp/Windows-Forms/Grundlagen>

<https://www.w3schools.com/cs/index.php>

<https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/tutorials/>

<https://www.tutorialspoint.com/csharp/index.htm>