

Aufgabe WindowsFormsSpielerInfothek

In dieser Aufgabe soll ein eine einfache WindowsForms SpielerInfothek als kleines Projekt realisiert werden. Für jeden Verein sollen die zugehörigen Spieler mit deren Information angezeigt, hinzugefügt und geändert werden können. Vereine sollen ebenfalls hinzugefügt und geändert werden können.

Legen Sie ein neues Projekt **WindowsFormsSpielerInfothek** an.
Designen Sie die **Oberfläche**. Denken Sie dabei an den späteren Benutzer.
Eine Variante ist unten dargestellt.

The screenshot shows a Windows Forms application window titled "WindowsFormsSpielerInfothek". The window is divided into two main sections: "Verein" (Club) and "Spieler" (Player).

Verein Section:

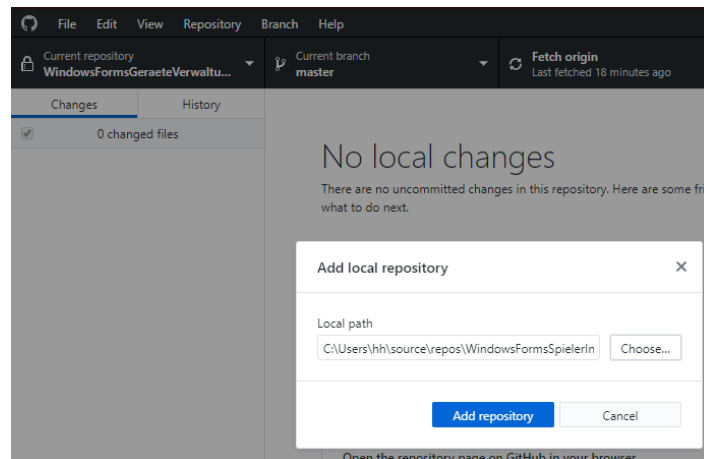
- Input fields: Id, Vereinsname, Kurzbezeichnung, Gegründet.
- Buttons: Verein ändern, Verein hinzufügen.

Spieler Section:

- Input fields: Id, Verein (dropdown), Vorname, Nachname, Geburtstag, Funktion (dropdown).
- Buttons: Spieler ändern, Spieler hinzufügen.

Fügen Sie das Visual-Studio-Projekt zur Git-Versionsverwaltung hinzu (Datei→Zur Versionsverwaltung hinzufügen).

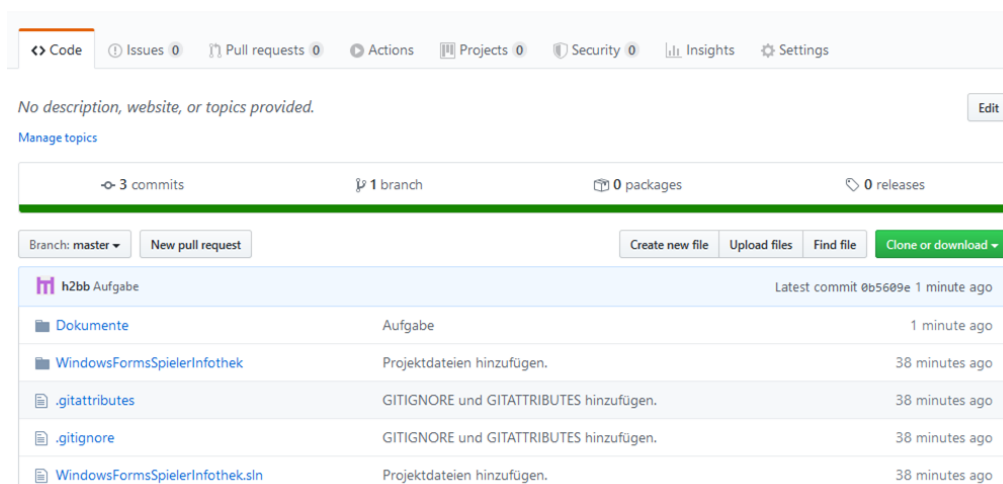
In **Github-Desktop** fügen Sie das Projekt als neues lokales Repository hinzu (File→Add local repository) und legen es anschließend als Remote-Repository auf dem Github-Server an (publish repository).



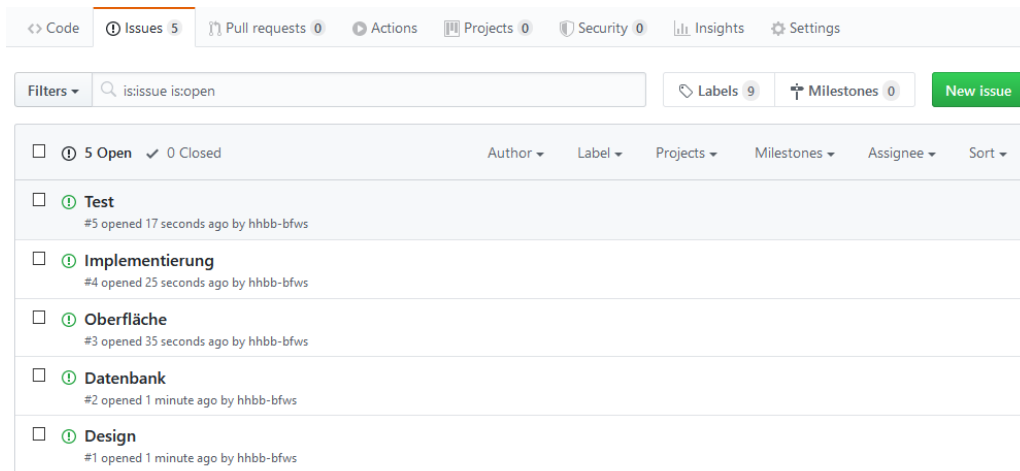
Wechseln Sie in das Projekt-Verzeichnis und legen ein Verzeichnis **Dokumente** an, in das Sie alle zum Projekt notwendigen Dokumente wie SQL-Skripte, Visio-Design-Dokumente, usw. legen. Da es sich im Projekt-Verzeichnis befindet, stehen die Dokumente ebenfalls unter der Versionsverwaltung.

Name	Änderungsdatum	Typ	Größe
.git	06.05.2020 05:36	Dateiordner	
.vs	06.05.2020 04:13	Dateiordner	
Dokumente	06.05.2020 05:43	Dateiordner	
WindowsFormsSpielerInfothek	06.05.2020 05:11	Dateiordner	
.gitattributes	06.05.2020 05:13	Textdokument	3 KB
.gitignore	06.05.2020 05:13	Textdokument	6 KB
WindowsFormsSpielerInfothek.sln	06.05.2020 04:13	Visual Studio Solu...	2 KB

Öffnen Sie das Repository auf dem **Github-Server**



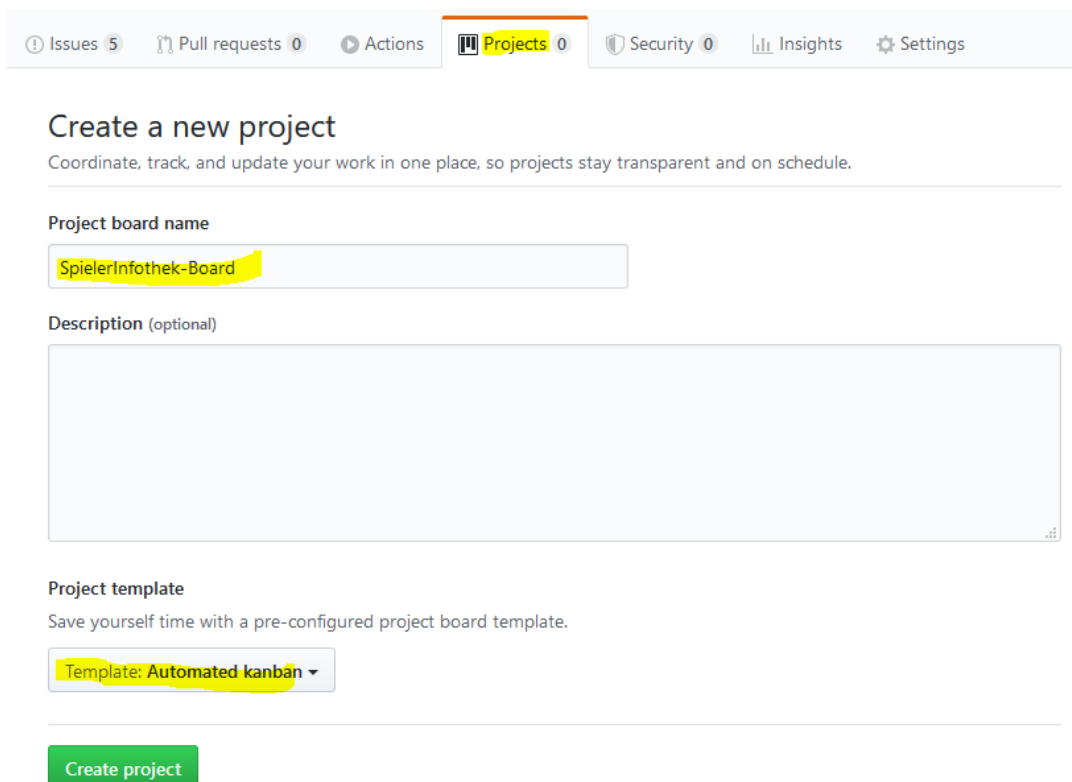
Legen Sie neue **Issues** (Tickets) an z.B. für Anforderungen, Design, Datenbank, Oberfläche, Implementierung, Test.



The screenshot shows the GitHub Issues interface. The top navigation bar includes links for Code, Issues (5), Pull requests (0), Actions, Projects (0), Security (0), Insights, and Settings. Below the navigation bar, there is a search bar with the filter 'is:issue is:open' and buttons for Labels (9) and Milestones (0). A green 'New issue' button is on the right. The main content area displays a list of 5 open issues, each with a checkbox, a green circle icon, a title, and a timestamp. The issues are: Test (#5, opened 17 seconds ago), Implementierung (#4, opened 25 seconds ago), Oberfläche (#3, opened 35 seconds ago), Datenbank (#2, opened 1 minute ago), and Design (#1, opened 1 minute ago). Each issue is followed by the text 'by hhbb-bfws'.

Issue	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/> ⑤ Test #5 opened 17 seconds ago by hhbb-bfws						
<input type="checkbox"/> ④ Implementierung #4 opened 25 seconds ago by hhbb-bfws						
<input type="checkbox"/> ③ Oberfläche #3 opened 35 seconds ago by hhbb-bfws						
<input type="checkbox"/> ② Datenbank #2 opened 1 minute ago by hhbb-bfws						
<input type="checkbox"/> ① Design #1 opened 1 minute ago by hhbb-bfws						

Legen Sie ein neues **Projekt** mit SpielerInfothek-Board mit Basic-Kanban-Template an.



The screenshot shows the 'Create a new project' form in GitHub. The top navigation bar includes links for Issues (5), Pull requests (0), Actions, Projects (0), Security (0), Insights, and Settings. The main heading is 'Create a new project' with a subtitle 'Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.' Below the heading, there is a section for 'Project board name' with a text input field containing 'SpielerInfothek-Board'. Below this is a section for 'Description (optional)' with a large text area. Below the description is a section for 'Project template' with a subtitle 'Save yourself time with a pre-configured project board template.' and a dropdown menu showing 'Template: Automated kanban'. At the bottom of the form is a green 'Create project' button.

Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name

SpielerInfothek-Board

Description (optional)

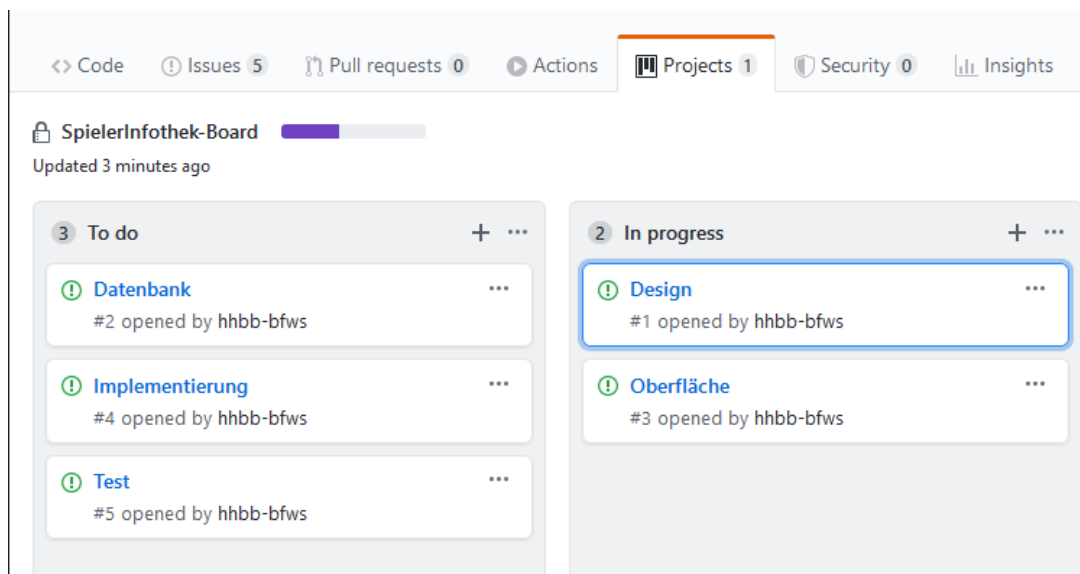
Project template

Save yourself time with a pre-configured project board template.

Template: Automated kanban

Create project

Schieben Sie die **Tickets** in die To-Do-Spalte.



Später können Sie in Github-Desktop die **Commits** in einfacher Weise einem Ticket zuordnen, indem Sie nach Eingabe von # in der Commit-Message-Zeile ein Ticket auswählen.

Anforderungen

Konkretisieren Sie die Anforderungen an Ihre Anwendung und erstellen ein **UML Anwendungsfall-Diagramm** mit den gewünschten Anwendungsfällen.

Hier z.B. Verein anzeigen, Spieler anzeigen, Verein hinzufügen, Spieler hinzufügen, Spieler ändern oder Verein des Spielers ändern.

Design

Erstellen Sie ein **ER-Diagramm** und bestimmen Entitäten, Attribute und Kardinalitäts-Beziehungen.

Erstellen Sie ein **UML Klassendiagramm** und beschreiben die notwendigen Klassen.

Entwerfen Sie die **Oberfläche**.

WindowsFormsSpielerInfothek

Verein

Id: 2, Vereinsname: RB Leipzig, Kurzbezeichnung: RBL, Gegründet: 20.05.2009

Spieler

Id: 4, Verein: RB Leipzig, Vorname: Timo, Nachname: Werner, Geburtstag: 06.03.1996, Funktion: Angriff

Id	Vereinsname	Kurzbezeichnung	Gegründet
1	Borussia Dortmund	BVB	01.01.1909
3	FC Bayern München	FCB	19.02.1900
2	RB Leipzig	RBL	20.05.2009

Id	VereinId	Vorname	Nachname	Geburtsdag	Funktion
6	2	Peter	Gulacsi	06.05.1990	Torwart
5	2	Lukas	Klostemann	03.06.1996	Abwehr
4	2	Timo	Werner	06.03.1996	Angriff

Implementierung

Implementieren Sie Teile der Anwendung entsprechend dem gewählten Anwendungsfall. Also beginnend mit Vereine anzeigen, Spieler anzeigen, usw.

Erstellen Sie die **Datenbank** mit den notwendigen Tabellen und füllen diese mit Testdaten, z.B. von <https://www.fussballdaten.de/bundesliga/>

Jede Tabelle erhält eine erste Spalte id mit Auto-Inkrement.

Keine Umlaute in Spaltennamen verwenden!

SQL ist Case **ins**ensitiv.

Erstellen Sie notwendigen **Daten-Klassen** mit Attributen, die den Tabellen und deren Spalten entsprechen.

(viele OR-Mapper machen Gebrauch von ‚Convention before Configuration‘)

Implementieren Sie die notwendigen **Ereignisbehandlungs-Methoden** für die vorgesehenen Schaltflächen (Buttons) und andere Steuerelemente (DataGridView, ComboBox).

Beginnen Sie mit dem Anwendungsfall Vereine anzeigen, Spieler anzeigen,...

Test

Testen Sie die Anwendung

Hinweis

- Soll NPoco als OR-Mapper verwendet werden, dann an die folgenden Aktionen denken:
PM> install-package npoco, Verweis auf MySql.Data hinzufügen, using NPoco hinzufügen, Connectionstring in App.config
Hinweise in ORM-Npoco.ppt und Aufgabe TelefonbuchNPoco

Daten-Klassen anlegen

```
class Verein
{
    public int Id { get; set; }
    public string Vereinsname { get; set; }
    public string Kurzbezeichnung { get; set; }
    public DateTime Gegrundet { get; set; }

    public override string ToString()
    {
        return $"{Vereinsname}";
    }
}

class Spieler
{
    public int Id { get; set; }
    public int VereinId { get; set; }
    public string Vorname { get; set; }
    public string Nachname { get; set; }
    public DateTime Geburtstag { get; set; }
    public string Funktion { get; set; }

    public override string ToString()
    {
        return $"{Vorname} {Nachname}";
    }
}
```

Verein anzeigen mit Fetch<Verein>

```
private void VereinLaden()
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            lstVerein.Clear();
            lstVerein = db.Fetch<Verein>("order by vereinsname");

            gridVerein.DataSource = null;
            gridVerein.DataSource = lstVerein;

            cboVerein.DataSource = null;
            cboVerein.DataSource = lstVerein;

            gridVerein_CellClick(this, new DataGridViewCellEventArgs(0, 0));
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```

**Eine Zeile (Verein) im gridVerein auswählen (Event CellClick),
Textboxen mit den Vereinsdaten aktualisieren und
zum Verein gehörende Spieler anzeigen**

```
private void gridVerein_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (gridVerein.SelectedRows.Count > 0)
    {
        Verein verein = gridVerein.SelectedRows[0].DataBoundItem as Verein;
        if (verein != null)
        {
            txtVereinId.Text = verein.Id.ToString();
            txtVereinsname.Text = verein.Vereinsname;
            txtKurzbezeichnung.Text = verein.Kurzbezeichnung;
            txtGegrundet.Text = verein.Gegrundet.ToShortDateString();

            SpielerLadenFür(verein.Id);

            gridSpieler_CellClick(sender, e);
        }
    }
}

private void SpielerLadenFür(int vereinId)
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            lstSpieler.Clear();
            lstSpieler = db.Fetch<Spieler>("where vereinId = @0 order by nachname",
                                           vereinId);

            gridSpieler.DataSource = null;
            gridSpieler.DataSource = lstSpieler;
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```

Eine Zeile (Spieler) im gridSpieler auswählen (Event CellClick) und die Textboxen füllen

```
private void gridSpieler_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (gridSpieler.SelectedRows.Count > 0)
    {
        Spieler spieler = gridSpieler.SelectedRows[0].DataBoundItem as Spieler;
        if (spieler != null)
        {
            txtSpielerId.Text = spieler.Id.ToString();
            txtVorname.Text = spieler.Vorname;
            txtNachname.Text = spieler.Nachname;
            txtGeburtstag.Text = spieler.Geburtstag.ToShortDateString();
            cboFunktion.Text = spieler.Funktion;
        }
    }
}
```

Verein hinzufügen

```
private void btnVereinHinzufügen_Click(object sender, EventArgs e)
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            Verein verein = new Verein();
            verein.Id = 0; // wird von DB gesetzt (autoincrement)
            verein.Vereinsname = txtVereinsname.Text;
            verein.Kurzbezeichnung = txtKurzbezeichnung.Text;
            verein.Gegrundet = Convert.ToDateTime(txtGegrundet.Text).Date;

            object o = db.Insert(verein);

            VereinLaden();
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```

Verein ändern

```
private void btnVereinÄndern_Click(object sender, EventArgs e)
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            Verein verein = new Verein();
            // id übernehmen, da update
            verein.Id = Convert.ToInt32(txtSpielerId.Text);
            verein.Vereinsname = txtVereinsname.Text;
            verein.Kurzbezeichnung = txtKurzbezeichnung.Text;
            verein.Gegrundet = Convert.ToDateTime(txtGegrundet.Text).Date;

            int n = db.Update(verein);

            VereinLaden();
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```


Spieler hinzufügen

```
private void btnSpielerHinzufügen_Click(object sender, EventArgs e)
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            Spieler spieler = new Spieler();
            spieler.Id = 0; // wird von DB gesetzt (autoincrement)
            Verein verein = cboVerein.SelectedItem as Verein;
            if (verein != null)
                spieler.VereinId = verein.Id;
            else
                spieler.VereinId = 0; // nicht vorhanden
            spieler.Vorname = txtVorname.Text;
            spieler.Nachname = txtNachname.Text;
            spieler.Geburtstag = Convert.ToDateTime(txtGeburtstag.Text).Date;
            spieler.Funktion = cboFunktion.Text;

            object o = db.Insert(spieler);

            SpielerLadenFür(verein.Id);
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```

Spieler ändern

```
private void btnSpielerÄndern_Click(object sender, EventArgs e)
{
    try
    {
        using (IDatabase db = new Database("mariadb"))
        {
            Spieler spieler = new Spieler();
            // id übernehmen, da update
            spieler.Id = Convert.ToInt32(txtSpielerId.Text);
            Verein verein = cboVerein.SelectedItem as Verein;
            if (verein != null)
                spieler.VereinId = verein.Id;
            else
                spieler.VereinId = 0; // nicht vorhanden
            spieler.Vorname = txtVorname.Text;
            spieler.Nachname = txtNachname.Text;
            spieler.Geburtstag = Convert.ToDateTime(txtGeburtstag.Text).Date;
            spieler.Funktion = cboFunktion.Text;

            int n = db.Update(spieler);

            SpielerLadenFür(verein.Id);
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```