Kenndaten

Sunday, March 19, 2023 12:36 AM

Funktionalitäten

Feature	Umsetzung	Benutzbar durch
Schützen-Liste		
Schützen auflisten	Vertikale Liste der Nutzer (Main-View)	Starten der App
Geburtstag anzeigen	Geschenk-Symbol in der Liste	Anschauen der Liste
Regelmäßige Teilname anzeigen	Grüner-Haken-Symbol in der Liste	Anschauen der Liste oder Edit-View
Schützen bearbeiten	Formular-Ansicht zu Nutzerdaten (Edit-View)	Antippen des Nutzers in Liste
Schützen anlegen	Formular-Ansicht zu Nutzerdaten (Edit-View)	Antippen des "Neuer Schütze"-Buttons
Schützen-Edit		
Name, E-Mail, GB setzen	Text-Bearbeitungs-Felder und Date-Picker	Antippen der Felder
Ein-/Austragen von Abteilungen & Trainingstragen	Vertikale Liste zur Darstellung Picker und +/- Buttons	Benutzen der Picker und Buttons
Gebühr anzeigen	Label in Edit-View	Bearbeiten der Abteilungen/ Ansehen des Labels

Team

Justin Mc Clain - Full Stack

Vorgehensweise

- Umgebung ermitteln
  - Stack wählen
    - DBMS ermitteln (SQLITE)
    - Verfügbare Datenbank-Libraries bei Stack-Wahl berücksichtigen
      - ORM?
      - Vereinfachungen mit Code? z.B. Code-Variablen Queriebar anstatt jede Query selbst zu schreiben (Bsp.: "AllUsers.Where(u => !u.Active)" bei uns mit LINQ möglich)
    - MVC-System?
    - Standalone oder Web-App?

        - Sicherheitsfragen beachten (compiled/interpreted language, online Daten-Transfer?)
    - Datenbankstruktur entwerfen
        - Benötigte Spalten und Beziehungen ermitteln
        - ER-Modell anlegen
        - Tables, Spalten und evtl. Contraints setzen (via ORM)
- Prototypisieren
    - Referenzen zu ähnlichen Visuals/Code recherchieren
    - Benötigte Dokumentation zur Umsetzung der Funktionalitäten im Bezug auf den Stack recherchieren
    - Datenbank-Konnektivität testen, Tabellen erstellen (lassen)
    - ORM-Klassen deklarieren
        - Passende Datentypen ermitteln
        - Beziehungen deklarieren/Implementieren
    - Grobes UI erstellen
    - Features grob und simpel einbauen
        - Queries/Beziehungen umsetzen
        - Funktionen deklarieren und implementieren
- Polish
    - Bestmögliche UI-Elemente wählen Position anpassen (evtl. Anpassen für Mobile)
    - Gründliches Testing und Fehlersuche
    - Code-Optimierungen
    - Code-Cleanup
- Export
    - Release-Config
    - Metadaten
    - Multi-Platform
    - Distribution

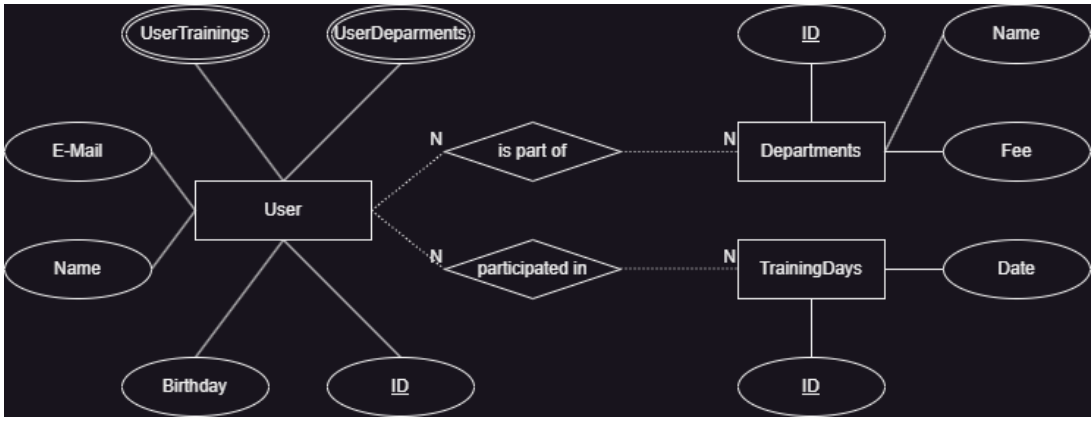
Datenbankstruktur

Sunday, March 19, 2023 1:39 AM

Schema

Name	Type
Tables (6)	
Department	
ID	integer
Fee	integer
Name	varchar
TrainingDays	
ID	integer
Date	bigint
User	
ID	integer
Name	varchar
Email	varchar
Birthday	bigint
UserDepartments	
UserID	integer
DepartmentID	integer
UserTrainings	
UserID	integer
TrainingDayID	integer

ER-Modell



App-Struktur/Stack

Sunday, March 19, 2023 11:33 AM

.NET MAUI

Multi-Platform **A**pp-**UI**

- **Entwickelt von Microsoft und Released Anfang 2022**
  - **Release wurde kritisch gesehen da Beispiele/Features und Dokumentation teilweise bis heute mangelhaft sind**
    - **Aufgrunddessen ist Entwicklung etwas Riskant durch die wenigen Resourcen**
    - **Gewählt wurde es trotzdem um die Technologie zu lernen und entdecken**
- **Eine Codebase für gängige Endbenutzer Betriebssysteme (Windows, Mac, Android, iOS)**
- **Funktioneller Nachfolger von Xamarin.Forms**
  - **Benutzt Xamarin Markup-Language (XAML) für UI**

Benutzte Tools

- **IDE & Debugger:** Visual Studio 2022 (+Android Virtual Device)
- **DB Browser:** DB Browser for SQLite
- **Dokumentation:** draw.io, Excel, OneNote
- **Online Resourcen:** Microsoft Docs/GitHub
- **DB Integration:** sqlite-net-pcl (NuGet)
- **DB Integration (Relationships):** SQLite-Net Extensions (NuGet)

Beispielrechnung

Sunday, March 19, 2023 11:25 PM

Timetable

Aktion	**Dauer**
.NET MAUI erlernen/einrichten	**12h**
Dokumentation & App-/Datenbank-Entwurf und Modell	**3h**
XAML lernen/UI zusammenstellen	**5h**
Features implementieren/testen	**5h**
Model-View-Controller-Service lernen/implementieren	**4h**
UI für Mobile optimieren/testen	**2h**
SQLite-Net erlernen/einrichten	**3h**
SQLite-Net Extensions erlernen/einrichten	**2h**
RelayCommand-Wissen erweitern	**3h**
Summe	**39h**
Beispielrechnung	**39hx45€ =1.755€**

Code-Ausschnitte

Sunday, March 19, 2023 10:54 PM

ORM-Klasse (User/Schütze)

```
41 references
public class User
{
    [PrimaryKey, AutoIncrement]
    5 references
    public int ID { get; set; }

    /* Properties */

    4 references
    public string Name { get; set; }
    2 references
    public string Email { get; set; }

    [NotNull]
    3 references
    public DateTime Birthday { get; set; } = DateTime.UtcNow;

    [Ignore]
    1 reference
    public bool HasBirthday { get; set; } = false;

    [Ignore]
    1 reference
    public int Fee { get; set; } = -1;

    [Ignore]
    2 references
    public bool Active { get; set; } = false;


    /* Lists */

    [ManyToMany(typeof(UserDepartments))]
    6 references
    public ObservableCollection<Department> Departments { get; set; } = new ObservableCollection<Department>();

    [ManyToMany(typeof(UserTrainings))]
    8 references
    public ObservableCollection<TrainingDays> TrainingDays { get; set; } = new ObservableCollection<TrainingDays>();
}
```

- Ob Geburtstag ist
- Die Gebühr und
- Regelmäßig-Flag

werden nicht in die DB eingetragen, da Sie sowieso immer neu berechnet werden

Definition von Beziehungen

Users beziehen

```
2 references
public async Task<List<User>> GetUsersAsync()
{
    await Init();
    AllDepartments = await GetDepartmentsAsync(); // We could also assing this in the Init() but maybe we want to add Departments In-App sometime
    var users = await _database.GetAllWithChildrenAsync<User>();
    foreach (User user in users)
    {
        UpdateUserLocals(user, getRegular: true);
    }

    return users;
}
```

<- Geburtstag, Gebühr... Lokal berechnen

Resultierende Queries:
Für Alle User:

- *select * from "User"*

Darauf dann pro User für Beziehungen:
Beispiel: User mit ID 1

- *select * from [Department] where [ID] in (select [DepartmentID] from [UserDepartments] where [UserID] = 1)*
- *select * from [TrainingDays] where [ID] in (select [TrainingDayID] from [UserTrainings] where [UserID] = 1)*

Regelmäßige Teilnahme ermitteln

Vorgabe:

- In den letzten 12 Monaten einmal im jeweiligen Monat anwesend
  oder
- 18x gesamt im letzten Jahr

```
1 reference
public static bool IsRegular(User usr)
{
    DateTime today = DateTime.Today;
    DateTime endDate = new DateTime(today.Year, today.Month, day: 1); // Start Last Month since the current is not usually not finished
    DateTime startDate = endDate.AddYears(-1); // 1 year back from Starting Date

    bool trained18Times = usr.TrainingDays.Count(tDay: TrainingDays => tDay.Date > startDate && tDay.Date <= endDate) > 18;

    if (trained18Times)
        return true;

    Debug.WriteLine(message: "----------------------");
    for (DateTime dt = startDate; dt <= endDate; dt = dt.AddMonths(1))
    {
        bool didTrainInMonth = usr.TrainingDays.Any(tDay: TrainingDays => tDay.Date.Month == dt.Month);
        Debug.WriteLine("{0:M/yyyy} - {1} - {2}", dt, usr.Name, didTrainInMonth);
        if (!didTrainInMonth)
        {
            Debug.WriteLine(message: "Inactive!");
            return false;
        }
    }

    return true;
}
```

^ 18x im Jahr trainiert? ^

< Jeden Monat der letzten 12 durchgehen und Anwesenheit suchen

Monat	Name	Anwesend in Monat?

```
----------------------
3/2022 - Rede - True
4/2022 - Rede - True
5/2022 - Rede - True
6/2022 - Rede - True
7/2022 - Rede - True
8/2022 - Rede - True
9/2022 - Rede - True
10/2022 - Rede - True
11/2022 - Rede - True
12/2022 - Rede - True
1/2023 - Rede - True
2/2023 - Rede - True
3/2023 - Rede - True
----------------------
3/2022 - Hey - False
Inactive!
```

Beispiel des Debug-Konsolen Outputs

<- Alle 12 Monate Anwesend

Gebühr ermitteln

```
1 reference
public static int GetFee(User usr)
{
    switch (usr.Departments.Count)
    {
        case 1:
            return usr.Departments[0].Fee; // get first&only Department Fee
        case > 1:
            return 20; // Fee for Multiple Departments TODO: PreProcessor Define
        default:
            return 0;
    }
}
```

Vorgabe:

- Jeweilige Abteilungs-Gebühr bei einer Abteilung
- 20€ bei mehreren

Trivial Facts

Sunday, March 19, 2023 12:36 AM

Features

Feature	Implementation	Usable by
Marksman-List		
List Marksmen	Vertical List of Users (Main-View)	Starting the App
Show Birthday	Present-Symbol in List-Element	Inspecting the List
Show regular Participation	Green Checkmark-Symbol in List-Element	Inspecting the List or Edit-View
Edit Marksman	Form for User-Data (Edit-View)	Tapping of User in List
Create Marksman	Form for User-Data (Edit-View)	Tapping "Neuer Schütze"-Button
Marksman-Edit		
Define Name, E-Mail, GB for User	Text-Edit Elements and Date-Picker	Tapping of Element
Ein-/Austragen von Abteilungen & Trainingstragen	Vertikale Liste zur Darstellung Picker und +/- Buttons	Usage of Picker and Buttons
Gebühr anzeigen	Label in Edit-View	Editing Departments/ Inspecting Label

Team

Justin Mc Clain - Full Stack

Course of Action

- Inspect & Determine Environment
  - Choose Stack
    - Choose DBMS (SQLITE)
    - Regard available Database-Libraries when choosing
      - ORM?
      - Easier DB-Access through Code? e.g. Queryable Code-Variables instead of write whole Queries (e.g. "AllUsers.Where(u => !u.Active)" we can do this using LINQ)
    - MVC-System?
    - Standalone or Web-App?
    - Regard Security-related Questions (compiled/interpreted language, online Data-

Transfer?)
    - Design Database Structure
        - Define required Rows and Relationships
        - Create ER-Diagram
        - Set Tables, Spalten and Contraints if needed (here via ORM)
    - Start Protocol
- Prototyping
    - Lookup References to similiar Visuals/Code or even whole Projects
    - Lookup Dokumentation to implement Features in Regards to the Stack
    - Check Database-Connectivity, Create Tables/Let Code create them
    - Declare ORM-Classes
        - Choose & Define required Datatypes
        - Declare/Implement required Relations
    - Prototype UI/Interaction Elements
    - Prototype Feature-Set
        - Handle Queries/Relationships
        - Declare/Implement Functions
- Polish
    - Choose best Possible UI-Elemente and adjust Positions (adapt to mobile if necesseray)
    - Thorough Testing and Bughunt
    - Code-Optimizations
    - Code-Cleanup
- Export
    - Release-Config
    - Metadata
    - Multi-Platform
    - Distribution

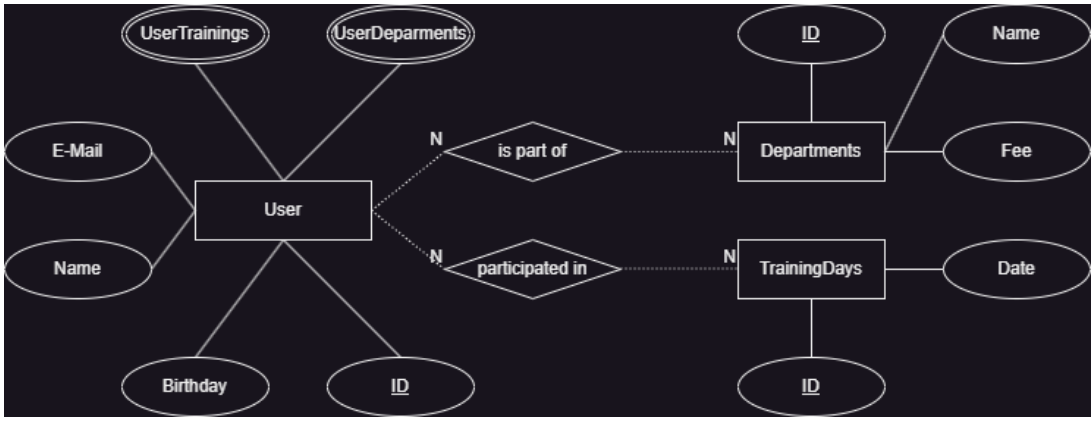
Database Structure

Sunday, March 19, 2023 1:39 AM

Schema

Name	Type
Tables (6)	
Department	
ID	integer
Fee	integer
Name	varchar
TrainingDays	
ID	integer
Date	bigint
User	
ID	integer
Name	varchar
Email	varchar
Birthday	bigint
UserDepartments	
UserID	integer
DepartmentID	integer
UserTrainings	
UserID	integer
TrainingDayID	integer

ER-Model



App-Structure/Stack

Sunday, March 19, 2023 11:33 AM

.NET MAUI

Multi-Platform **A**pp-**UI**

- **Developed by Microsoft and Released Early 2022**
  - **Release criticized for not having enough Features/Examples/Docs, this problem partially persists until today**
    - **Development for this might bring many Hurdles, based on the limited Learning-Resources available**
    - **It was chosen nonetheless to explore and get a feel for the Technology**
- **Promises to let you have one Codebase for all common End-User Operating-Systems (Windows, Mac, Android, iOS)**
- **Unoffically called Sequel to Xamarin Forms**
  - **Uses Xamarin Markup-Language (XAML) for designing UI**
- **Only usable in Visual Studio 2022 using .NET C#**

Tools used

- **IDE & Debugger:** Visual Studio 2022 (+Android Virtual Device)
- **DB Browser:** DB Browser for SQLite
- **Documentation:** draw.io, Excel, OneNote
- **Online Resources:** Microsoft Docs/GitHub
- **DB Integration:** sqlite-net-pcl (NuGet)
- **DB Integration (Relationships):** SQLite-Net Extensions (NuGet)

Example Calculation

Sunday, March 19, 2023 11:25 PM

Timetable

Action	**Duration**
Learn/Get going with .NET MAUI	**12h**
Documentation & App-/Database-Design and Model	**3h**
Learn XAML/Putting together UI	**5h**
Implement/Test Features	**5h**
Model-View-Controller-Service Implementation/Learning	**4h**
Test/Optimize UI for Mobile	**2h**
Learn/Get going with SQLite-Net	**3h**
Learn/Get going with SQLite-Net Extensions	**2h**
Expand RelayCommand-Knowledge	**3h**
Sum	**39h**
Example Calculation	**39hx45€ =1.755€**