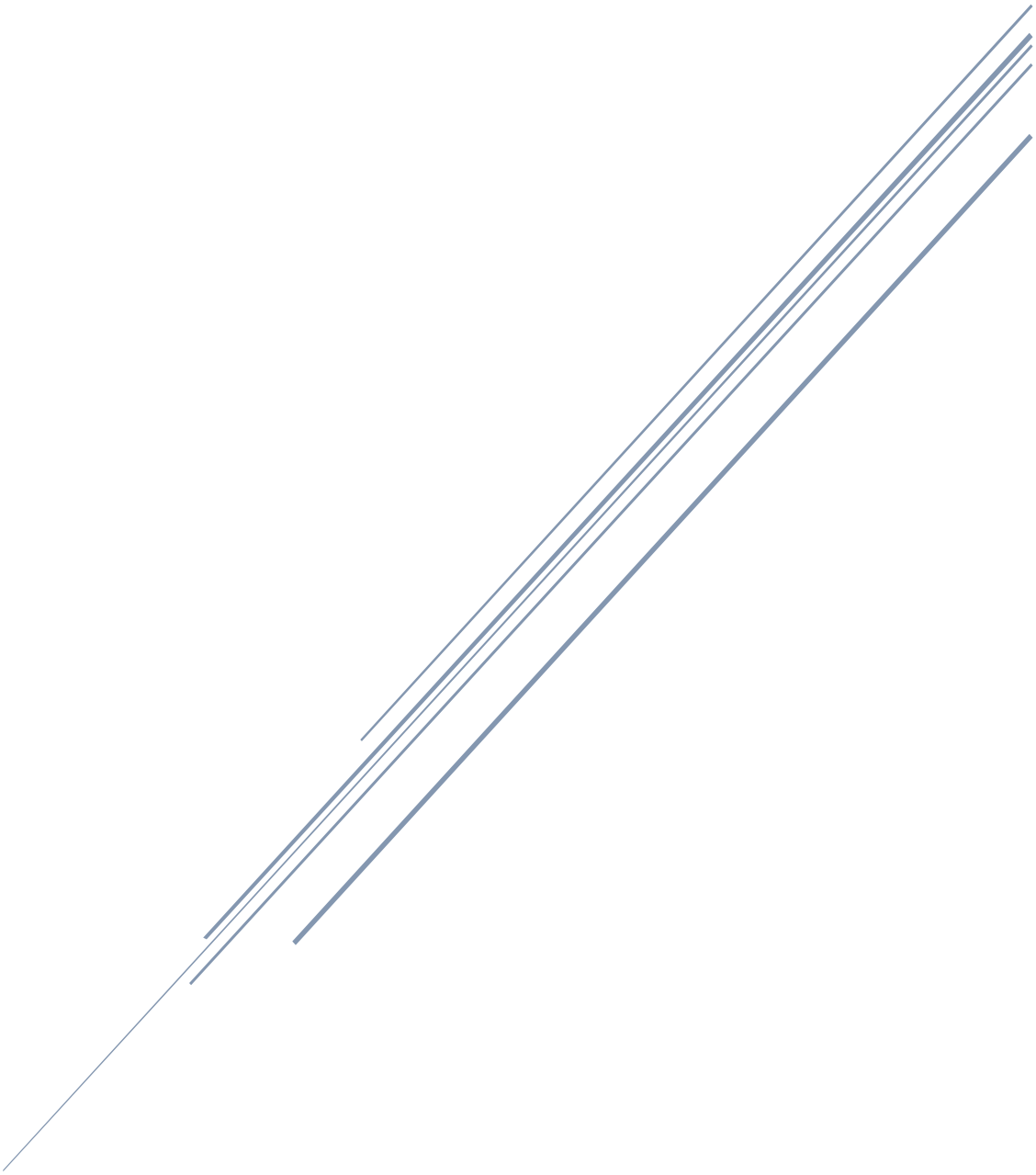
DOKUMENTATION

Stronk-M133



IMS Basel
Raphael Gisler

Inhalt

Einleitung

Diese Dokumentation dokumentiert alles was man wissen muss über Stronk. Stronk ist eine Web-Applikation, welche ich für das Modul 133 und für alle Menschen welche gerne Kraftsport treiben, gemacht habe. Leute aus aller Welt können sich auf dieser Website versammeln und ihre Lieblings Workouts miteinander teilen. Man kann seine eigenen Workouts und Übungen auf der Web-Applikation erstellen und diese mit den anderen Nutzern teilen.

Umfeld und Ablauf

Dieses Projekt ging vom 28.11.2022 bis 11.12.2022 und musste mit der Programmiersprache C# und SQL-Server als Datenbank gemacht werden. C# musste man nach MVC (Model View Controller) machen, um somit eine Web-Applikation zu erstellen. Wenn man wollte, durfte man ebenfalls JavaScript benutzen. Ich habe JavaScript nur sehr kurz für das Frontend angewendet.

Informationen

Verwendete Sprachen

Der gesamte C# Code, die Datenbank und die Website selbst sind auf Englisch geschrieben. Da die Dokumentation aber auf Deutsch ist, ist es wichtig anzumerken, dass man sich nicht von dem Unterschied der Sprachen verwirren lassen sollte. Eine Übung wird im Code und in den Diagrammen als ‘Exercise’ Beschrieben und ist ein und dasselbe. Manche Wörter werden aber aus dem Englischen übernommen wie zum Beispiel das Wort ‘Workout’, ‘Post’ oder ‘User’, weil diese Wörter heutzutage auch im Deutschen verwendet werden. Ein Workout ist ein Training, ein Post ist ein Beitrag und ein User ist ein Benutzer.

Bedingungen

Die Art und Weise der Web-Applikation darf man fast frei wählen, aber es gibt gewisse Anforderungen, an welche man sich halten muss. Hier sehen Sie die verschiedenen Anforderungen aufgelistet.

Technische Anforderungen

- Web-Applikation mit Datenbank CRUD Operationen
- Visual Studio
- MS SQL-Server Datenbank
- C# ASP.Core MVC
- JavaScript/JQery

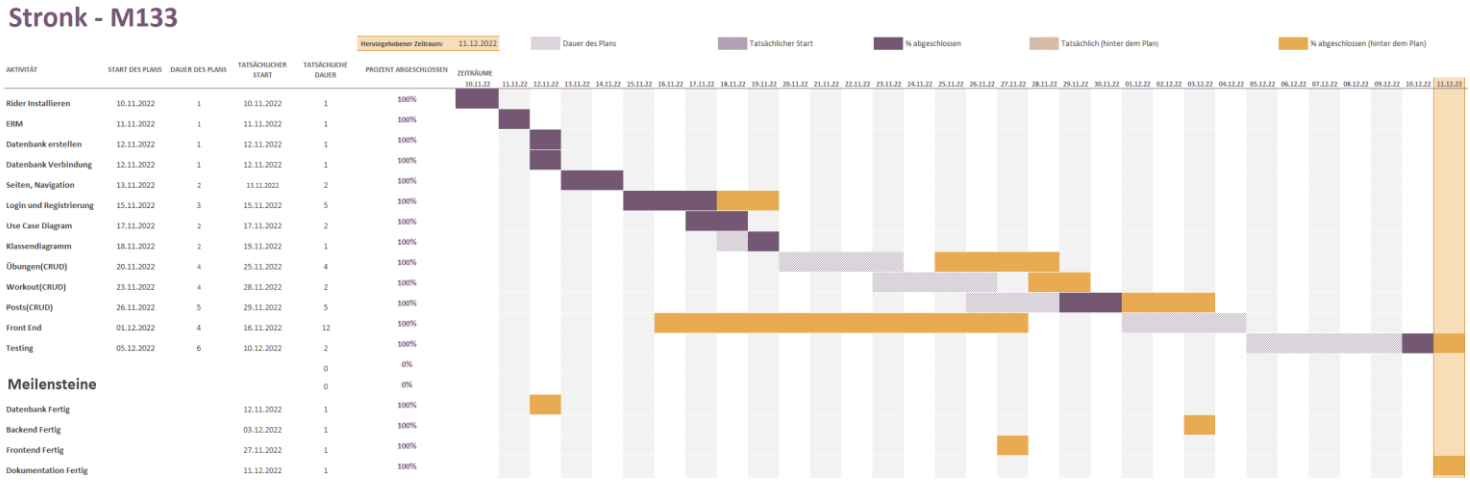
Funktionale Anforderungen

- 3 Tabellen mit Beziehungen mit zugehörigen CRUD-Funktionen (mit GUI)
- Benutzerverwaltung mit Rollenzuweisungen (ohne Verwaltungs-GUI)
- Datenschutz

Projektplan

Da es das erste Mal war, dass ich mit C# eine Web-Applikation gemacht habe, war es sehr schwer diesen Projektplan zu erstellen. Ich hatte besonders Probleme mit dem Einschätzen der CRUD Funktionen, da es etwas war, was mehr Wissen über das Entity Framework forderte. Der Hauptgrund für die vielen Verspätungen, waren wahrscheinlich, dass ich komplett unterschätzt habe wieviel Aufwand es ist, an das gesamte Wissen hinter C#, MVC, Entity Framework und die Verbindung der Datenbank zu kommen. In der letzte Woche mag es zwar so aussehen, als hätte ich nichts gemacht, aber die meiste Zeit war ich da am Debuggen.

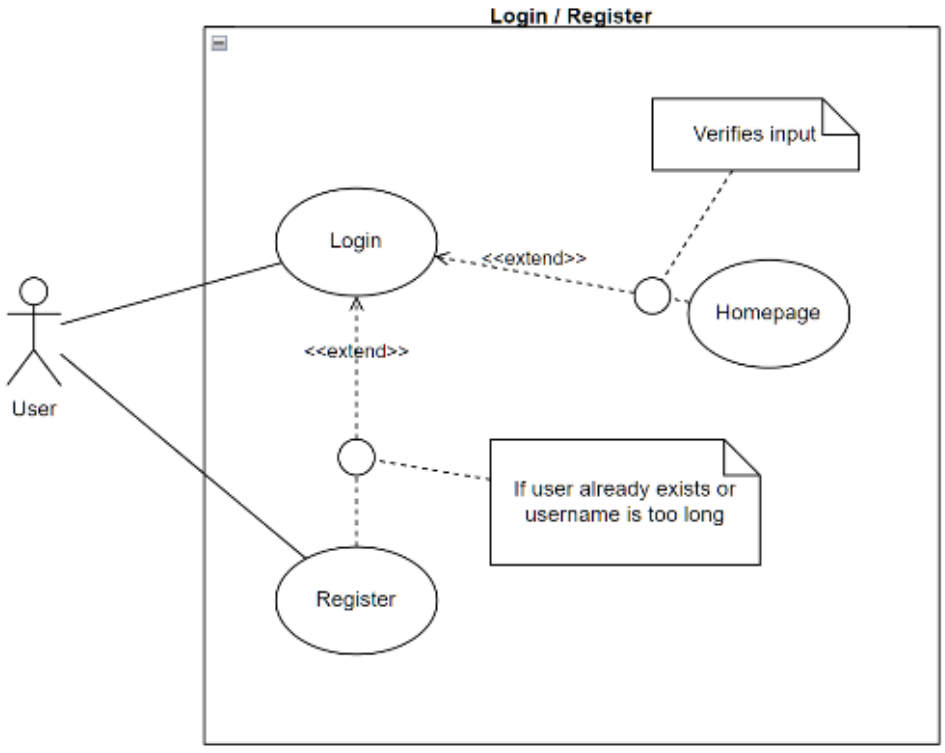
Ich glaube, wenn ich jetzt ein ähnliches Projekt mit C# nochmals machen müsste, würde mein Projektplan um einiges besser aussehen.



Use Case Diagram

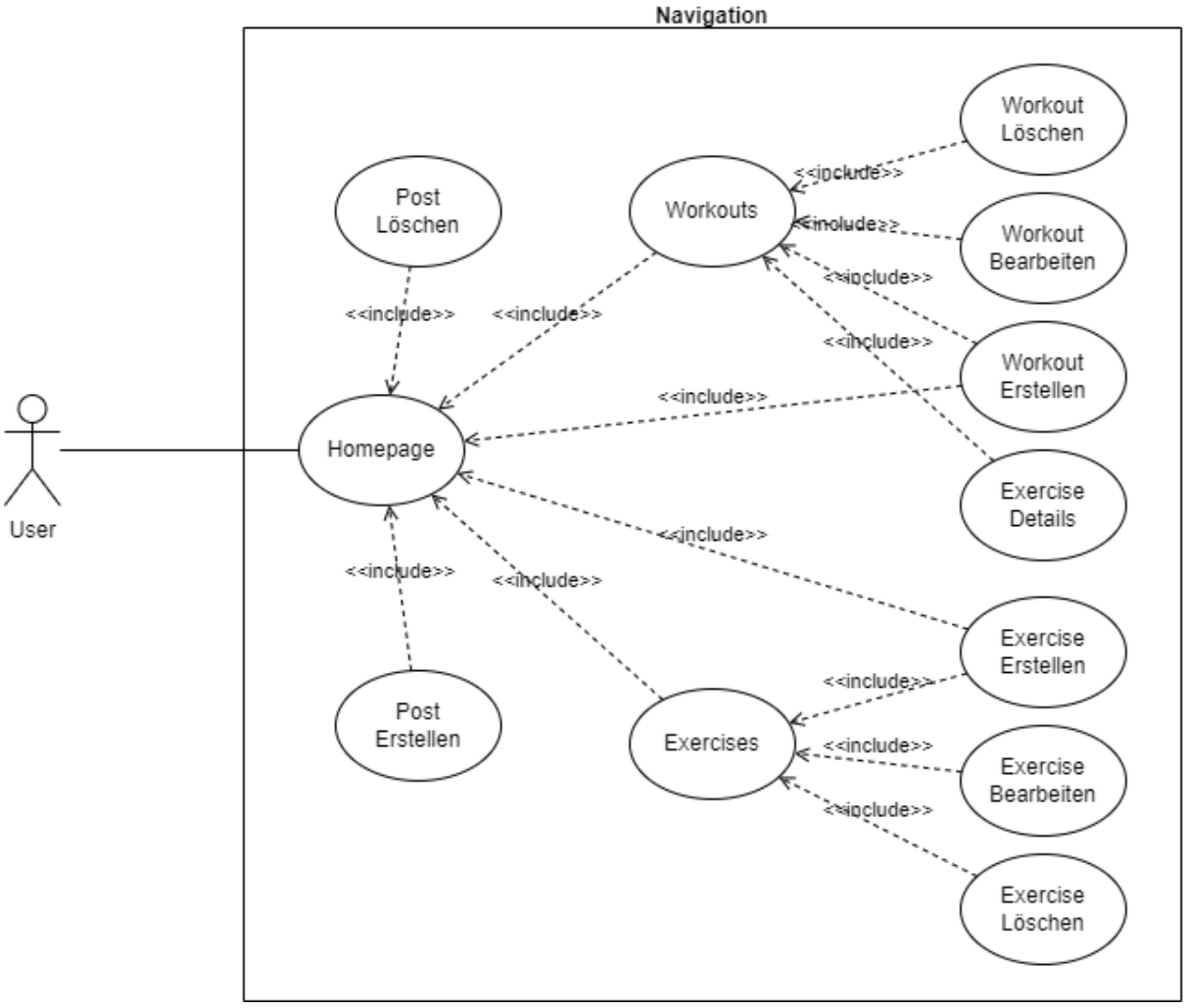
Login Beschreibung

Name	Login / Registrierung
Ziel	Eine Person kann sich auf der Website Registrieren, und Einloggen
Vorbedingung	- Eine Person, befindet sich auf der Website und hat noch kein Login gemacht. - Die Datenbank ist vollständig aufgestellt.
Nachbedingung	Ein neuer User wurde in der Datenbank erfasst und eine Person kann über diesen User auf die Website gelangen.
Akteure	User
Auslösendes Ereignis	Person, welche kein Login gemacht hat, ist auf der Website.
Beschreibung	1. Eine Person, welche sich nicht registriert hat, gelangt auf die Website. 2. Sie navigiert von dem Login zur Registrierungsseite. 3. Benutzername und Password werden ins Formular eingetragen und die Person drückt auf den Registrierungsknopf. 4. Ein neuer User wird in der Datenbank erstellt. 5. Die Person wird zur Login Seite gesendet. 6. Mit den registrierten Daten kann sich die Person nun einloggen, um auf die Homepage zu gelangen.
Alternativen	- Der gewählte Benutzername der Person ist bereits verwendet und es muss einen neuen gewählt werden. - Die Person ist bereits registriert und betätigt direkt das Einloggen. - Die Person entscheidet sich dagegen sich einzuloggen und verlässt die Website nach dem Registrieren. - Die Person gelangt auf die Website und verlässt sie wieder.

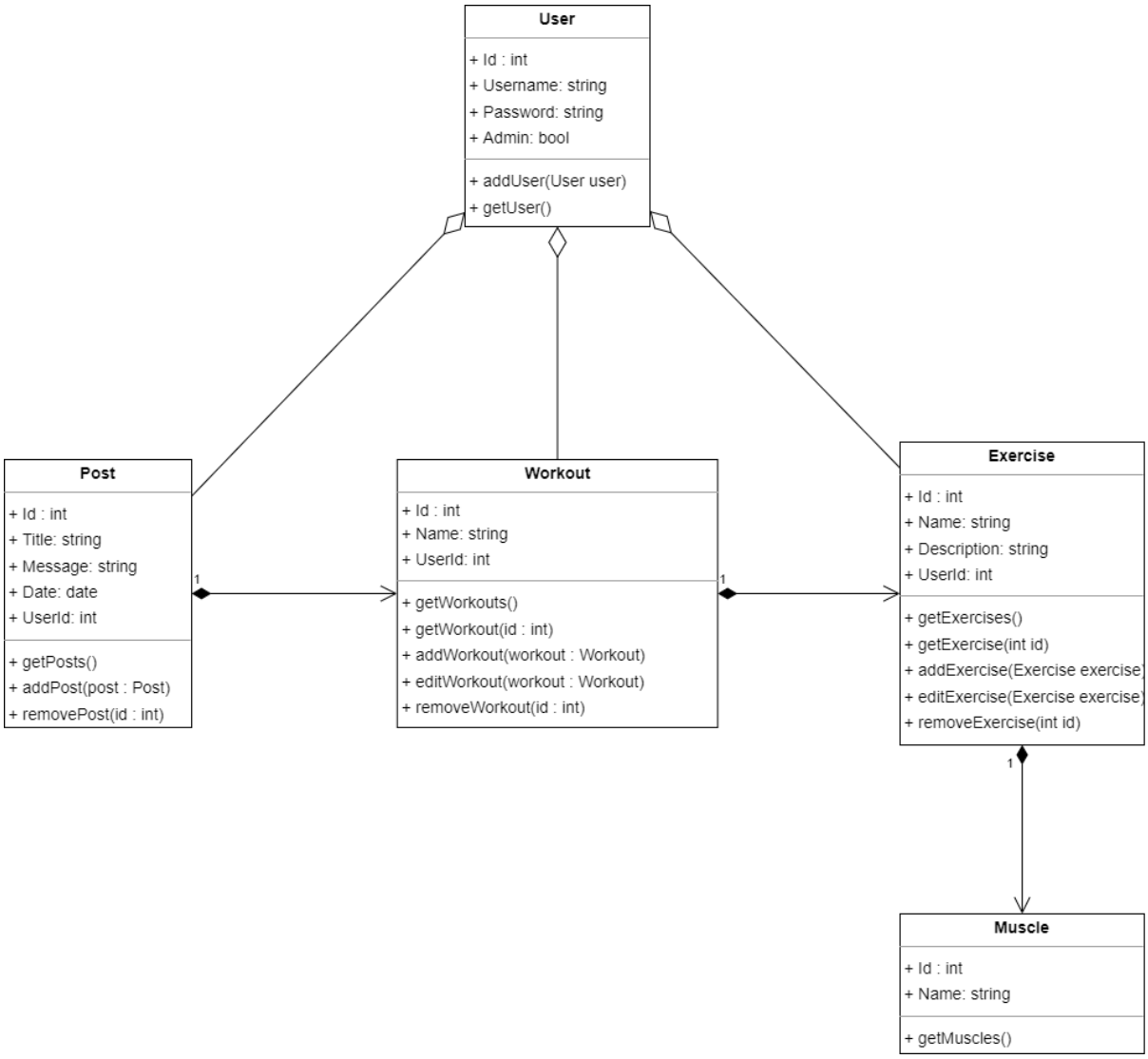


Navigation Beschreibung

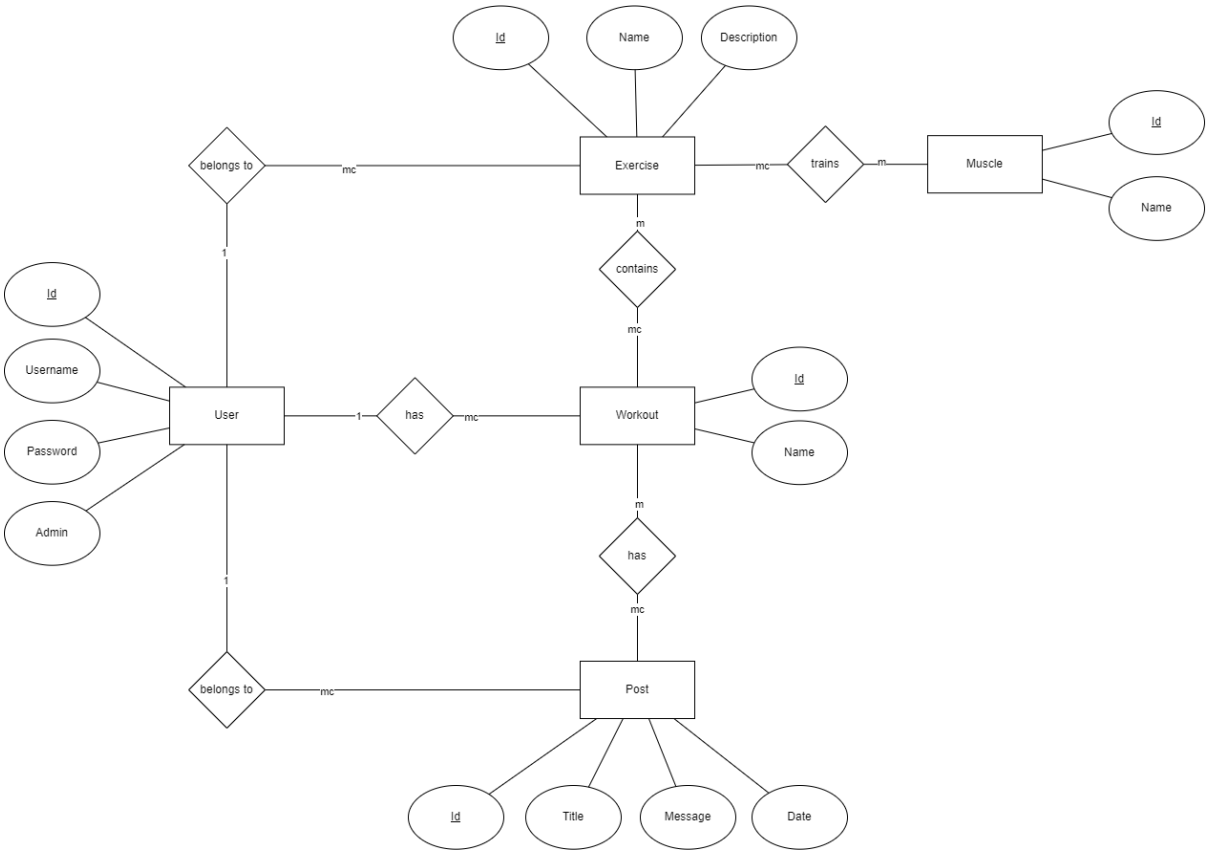
Name	Navigation
Ziel	Ein Benutzer/Admin kann die Website problemlos Navigieren und die Funktionen nutzen, welche sie zu bieten hat.
Vorbedingung	- Ein eingeloggter User befindet ist auf der Homepage
Nachbedingung	- Ein User hat die gesamte Website navigiert und kann die CRUD Funktionen, welche die Website zu bieten hat, nutzen.
Akteure	User
Auslösendes Ereignis	Benutzer logged sich ein
Beschreibung	1. Benutzer hat sich eingelogged und befindet sich auf der Website. 2. Benutzer erstellt oder löscht einer seiner Posts. 3. Benutzer navigiert zu den Workouts und verwaltet sie. 4. Benutzer navigiert zu den Übungen und verwaltet sie. 5. Benutzer logged sich aus.
Alternativen	- Der Benutzer kann sich ausloggen, bevor er sich die Website angeschaut hat.



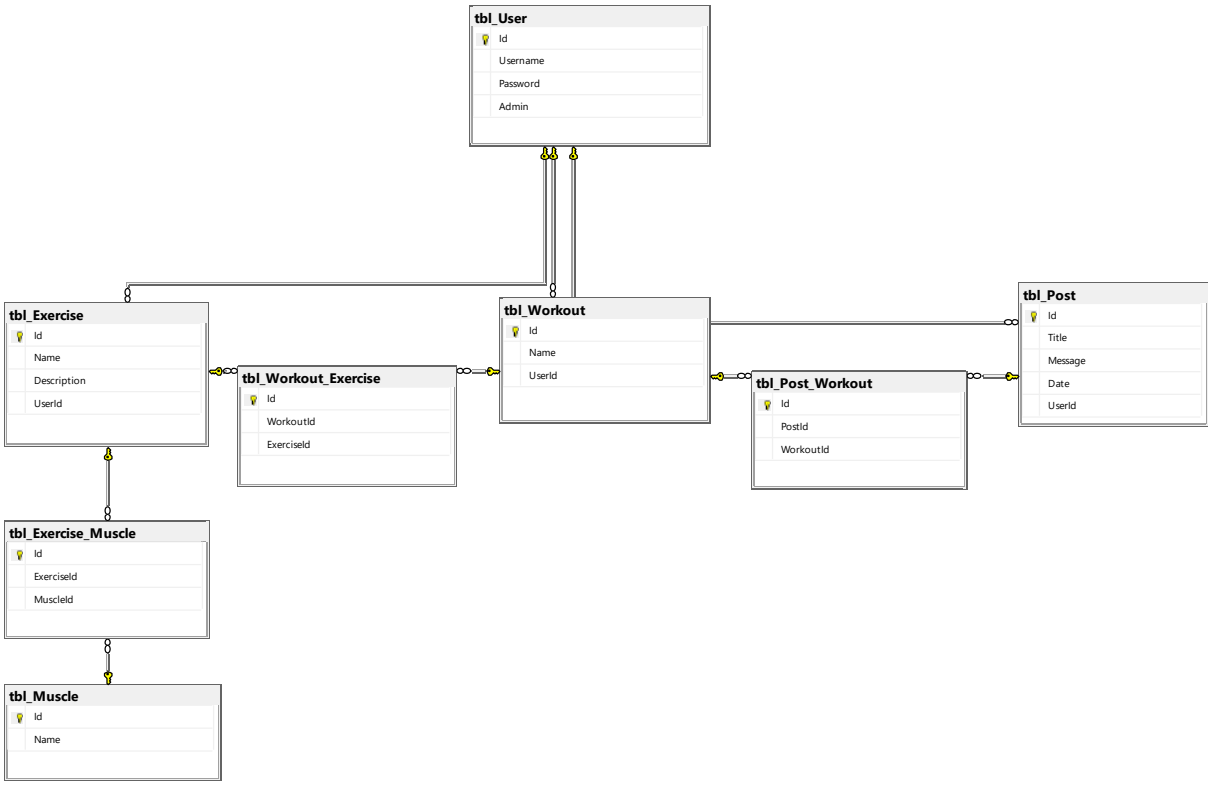
Klassendiagramm



ERM (Entity Relationship Model)



ERD (Entity Relationship Diagram)



Datenbank

Hier wird die Datenbank beschrieben. Es werden die wichtigen Tabellen und Ihre Spalten erklärt.

User

- **Username [NVARCHAR (90)]**
  Hier wird der Benutzername, welcher beim Registrieren eingetragen wurde, festgehalten. Der Benutzername wird benötigt, um sich einzuloggen und zu sehen, wem einem Post gehört. Der Benutzername darf maximal eine Zeichenlänge von 50 Zeichen beinhalten.
- **Password [CHAR (64)]**
  Eine Web-Applikation mit verschiedenen Benutzern benötigt Passwörter für die jeweiligen Benutzern. Aus Sicherheitsgründen dürfen sich Passwörter nicht so wie sie sind in der Datenbank befinden. Falls es zu einem Data breach kommen würde, könnten alle Passwörter zusammen mit den Benutzernamen an die Öffentlichkeit gelangen. Deswegen muss man Passwörter hashen oder encrypten. Ich habe mich für das hashen entschieden, weil meine Web-Applikation nicht die Funktion benötigt, die Passwörter wiederherzustellen.
  Ich habe mich für die SHA256 Algorithmus entschieden, da es einen optimalen Kompromiss zwischen Performance und Sicherheit hat. Eine weitere Option wäre SHA512 aber mit diesem Algorithmus benötigt man doppelt so viel Speicherplatz und es erhöht die Sicherheit nur ein wenig.
- **Admin [BIT]**
  Auf der Web-Applikation gibt es zwei verschiedene Benutzer. Es gibt den normalen Benutzer und den Administrator. Diese Spalte in der Tabelle zeigt an, ob ein Benutzer ein Admin ist oder nicht.

Post

Ein Post enthält ein oder mehrere Workouts.

- **Title [NVARCHAR (50)]**
  Hier kann der Inhalt des Posts kurz und knapp in dem Titel des Posts festgehalten werden.
- **Message [NVARCHAR(MAX)]**
  Hier kann der Benutzer eine Nachricht schreiben, welche bei einem Post angezeigt wird. Diese Nachricht sollte benutzt werden, um den Post und seine Workouts zu beschreiben.
- **Date [DATE]**
  Hier wird das Datum angezeigt, an welchem ein Post erstellt wird.

Workout

Ein Workout enthält ein oder mehrere Übungen.

- **Name [NVARCHAR (50)]**
  Der Name eines Workouts wird hier festgehalten. Ein Workout dient dazu, um mehrere Übungen zusammen zu nehmen, um somit eine klare Übersicht zu haben.

Exercise

Eine Übung enthält ein oder mehrere Muskelgruppen.

- **Name [NVARCHAR (50)]**
- **Description [NVARCHAR (MAX)]**
  Hier kann die Übung in weiterem Detail beschrieben werden. Oftmals kann man nur anhand von dem Namen nicht genau wissen worum es sich bei der Übung handelt. Deswegen hat

die Beschreiben einen NVRACHAR (MAX) erhalten, da man so mehr als genug Platz hat, um die Übung gut zu beschreiben.

Muscle

Eine Muskelgruppe kann in mehreren Übungen vorhanden sein.

- **Name [NVARCHAR (50)]**

Farben und Grafiken

Farben

Verwendete Ressourcen:

- Was ist Dark Mode und warum benutzen Leute ihn: https://www.axigen.com/articles/dark-mode-trend-benefits-history_121.html

Ich habe mich für ein Dark Mode entschieden, weil ich selbst merke, dass der Dark Mode in jüngeren Generationen immer wie mehr verwendet wird. Mir persönlich gefällt der Dark Mode vom Aussehen her auch einfach besser.

Hintergrund: 

Hellere Farbe: 

Schatten: 

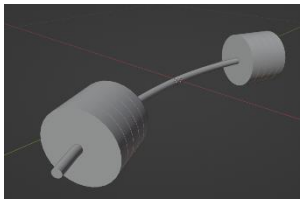
Es ist sehr einfach mit einem bestimmten Set von Farben zu arbeiten, weil man diese als CSS-Variablen im Root festhalten kann.

```
:root {
  --textColor: white;
  --bg: rgb(14 19 22);
  --light: rgb(29 37 43);
  --shadow: rgb(13 13 13);
}
```

Grafiken

Ich wollte mir mein eigenes Logo erstellen, weil ich ansonsten rechtliche Probleme haben könnte, wenn ich mir ein Bild aus dem Internet kopiert hätte.

Hier sehen sie das Logo. Das Logo selbst habe ich mit der 3D Software Blender gemacht. Ursprünglich wollte ich es mit einer 2D Software wie Photoshop

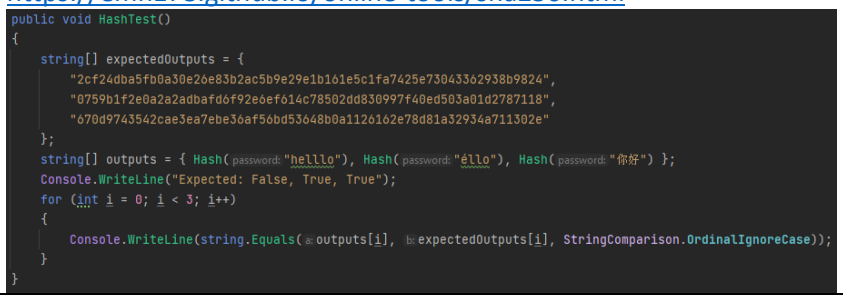


machen, weil ich es auch ein 2D Logo ist. Ich habe
aber schnell gemerkt, dass ich in 2D sehr schlecht bin.

Testen

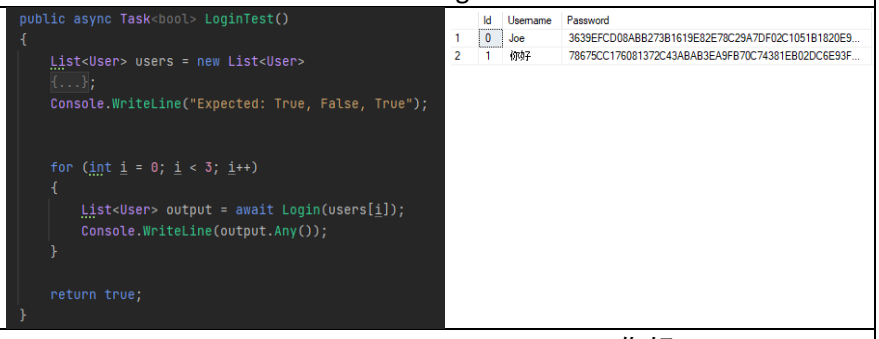
Alle Tests wurden in dem Browser Firefox getestet.

Hash

Was wurde getestet?	Ich habe getestet ob die Hash Funktion, welche mir meine Passwörter hashed wirklich funktioniert.
Wie wurde getestet?	Die Hashs welche bereites im Code vorhanden sind habe ich mit einem Hash Generator gemacht. https://emn178.github.io/online-tools/sha256.html 
Inputs	helllo, éllo, 你好
Erwartete Outputs	False, True, True
Outputs	False, True, True

```
public void HashTest()
{
    string[] expectedOutputs = {
        "2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824",
        "0759b1f2e0a2a2adbafd6f92e6ef614c78502dd830997f40ed503a01d2787118",
        "670d9743542cae3ea7ebe36af56bd53648b0a1126162e78d81a32934a711302e"
    };
    string[] outputs = { Hash(password: "helllo"), Hash(password: "éllo"), Hash(password: "你好") };
    Console.WriteLine("Expected: False, True, True");
    for (int i = 0; i < 3; i++)
    {
        Console.WriteLine(string.Equals(a: outputs[i], b: expectedOutputs[i], StringComparison.OrdinalIgnoreCase));
    }
}
```

Login

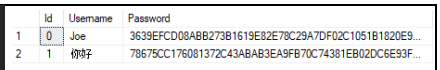
Was wurde getestet?	Es wurde getestet ob man sich mit den richtigen Logindaten einloggen kann und ob man mit den falschen abgelehnt wird.
Wie wurde getestet?	
Inputs	User1 = { Joe, Hi } , User2 = { ted, tester }, User3 = { 你好, joe }
Erwartete Outputs	True, False, True
Outputs	True, False, True

```
public async Task<bool> LoginTest()
{
    List<User> users = new List<User>
    {...};
    Console.WriteLine("Expected: True, False, True");


    for (int i = 0; i < 3; i++)
    {
        List<User> output = await Login(users[i]);
        Console.WriteLine(output.Any());
    }

    return true;
}
```

	Id	Username	Password
1	0	Joe	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E9...
2	1	你好	78675CC176081372C43ABAB3EA9FB70C74381EB02DC6E93F...

Register

Was wurde getestet?	Ob man sich registrieren kann und falls man einen Benutzername wählt, der bereits vorhanden ist, ob man abgelehnt wird.

Wie wurde getestet?	 
Inputs	User1 = { Joe, étest } , User2 = {你好, josh }, User3 = { Jon, Skywalker }
Erwartete Outputs	False, False, True
Outputs	False, False, True Nur der User3 wurde in die Datenbank eingefügt, da die anderen Benutzernamen bereits in der Datenbank vorhanden waren.

	Id	Username	Password
1	0	Joe	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99...
2	1	你好	78675CC176081372C43ABAB3EA9FB70C74381EB02DC6E93F...
3	2	Jon	90A34D07A3826B71EE32FD635F1BF6AB6374240AEAB4D3E4...

Arbeitsjournal

10.11.2022

Ich habe mich dazu entschieden, Rider von Jetbrains, als Editor zu verwenden. Ich habe mich gegen Visual Studio entschieden, da ich mit den Produkten von Jetbrains öfters gearbeitet habe und mir das Layout von Visual Studio wenig gefällt wie das von den Jetbrains Produkte.

11.11.2022

Ich habe es geschafft, die Datenbank mit C# zu verbinden. Dafür habe ich das 'System.Data.SqlClient' package heruntergeladen. Um die Verbindung herzustellen, benötigte ich einen Connection String. Dieser String beinhaltet unter anderem Username und Passwort des Datenbank Benutzers. Da meine Datenbank aber Windows Authentication benutzt, konnte ich diese zwei Parameter entfernen und sie durch 'Integrated Security=SSPI' ersetzt.

14.11.2022

Verwendete Ressourcen:

- Wie man ein Form in ASP.Net macht (YouTube Tutorial): https://youtu.be/3THbdVew2WI
- Artikel vom YouTube Tutorial: https://www.aspsnippets.com/Articles/ASPNet-MVC-Form-Submit-Post-example.aspx
- Microsoft Dokumentation über Hashbytes: https://learn.microsoft.com/en-us/sql/t-sql/functions/hashbytes-transact-sql?view=sql-server-ver16

Problem:
Ich wollte ein Form machen, welches an den Controller geschickt wird, um die Daten zu verwalten.

Im Parameter des Controllers war eine Model Instanz. Die Daten wurden aber nicht an den Parameter weitergegeben.

Lösung:
Ich dachte die ganze Zeit, das Problem wäre beim View oder Controller, es war aber beim Model. Ich habe vergessen, den Attributen des Models Getter und Setter zu geben. Somit konnte auch nicht auf die Attribute zugegriffen werden.

Ich habe eine Prozedur erstellt, um sich registrieren zu können. Ich musste lernen wie man in der Datenbank Hashs macht. Ich war mich nicht sicher welcher Algorithmus ich wählen sollte. Ich war mir nicht sicher ob ich 'MD5' oder 'SHA2_512' wählen sollte. Ich habe mich am Ende für 'SHA2_512'.

Für das Einloggen eine Prozedur erstellt, welches den Benutzernamen und das Passwort kontrollierten.

Ich habe für das Einloggen, in der Datenbank eine Prozedur erstellt, welche einen Username und Passwort als Input nimmt und es mit dem aus der Datenbank vergleicht. Das wird mir später helfen, weil ich weniger C# schreiben muss.

15.11.2022

Verwendete Ressourcen:

- Artikel, welcher sagt, dass BCRYPT besser ist: https://rietta.com/blog/bcrypt-not-sha-for-passwords/

Ich habe mich dazu entschieden einen Index auf meine Tabelle namens tbl_Muscle zu setzten. Diese Tabelle beinhaltet verschiedene Muskelgruppen. Benutzer der Webseiten können die Muskelgruppen nicht ändern oder neue erfinden. Sie müssen nur bei dem Erstellen einer neuen Übung ausgewählt werden können. Dies ist optimal für einen Index, weil der Index am besten funktioniert, wenn in der Tabelle viel abgefragt und wenig eingefügt werden muss.

Aufgrund weiterer Recherche ist mir aufgefallen, dass 'SHA2_512' doch nicht so gut ist, um Hashs von Passwörtern zu erstellen. Ich habe mich dazu entschieden BCRYPT zu verwenden. Das Problem war, dass das Erstellen von BCRYPT Hashs in SQL-Server nicht automatisch vorhanden ist.

16.11.2022

Verwendete Ressourcen:

- Wie man in C# BCRYPT Hash erstellt (YouTube Tutorial): https://youtu.be/5PYFUscoPCc

Da ich mich dazu entschieden habe BCRYPT als Hash zu verwenden, musste ich dies in C# selber und nicht in der Datenbank machen. Dazu habe ich mir ein kurzes YouTube Video angeschaut. Darauf habe ich mir ein NuGet Package namens 'BCrypt.Net' heruntergeladen und konnte damit ganz einfach Hashs von Passwörtern erstellen. Ich musste aber auch meine Login und Registrierung

Prozedur anpassen, da ich die Hashs nichtmehr in der Datenbank machte und sie da ebenfalls nicht mehr kontrollierte.

17.11.2022

Verwendete Ressourcen:

- Wie man Models erstellt (YouTube Tutorial): https://youtu.be/p2kzp2d0a4A?list=PL82C6-O4XrHde_urqhKJHH-HTUfTK6siO
- Microsoft Dokumentation über das Entity Framework: https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/implementing-basic-crud-functionality-with-the-entity-framework-in-asp-net-mvc-application

Um meine Use Cases zu erstellen, wollte ich mit einem einfacheren Anfangen, nämlich dem Login und Register. Dieser Prozessvorgang ist sehr kurz und konnte sehr stell dargestellt werden.

Nachdem wir im Unterricht über das Hashen von Passwörtern gesprochen haben, wurde mir klar, dass meine Passwörter gar nicht Encrypted sein müssen, sondern nur gehashed sein müssen. Deshalb habe ich mich wieder dazu umentschieden, auf meine alte Hash Methode zurück zu greifen, wo ich das Hashen und das Überprüfen, alles in der Datenbank machen würde. Somit bin ich von BCRYPT zurück auf SHA2_256 zurück gegangen.

Problem:
Ich weiss sehr wenig darüber, wie MVC mit C# funktioniert.

Um mir die Grundbausteine zu geben, um das Login und Register zu machen, habe wollte ich die Models erstellen. Dafür habe ich die Struktur meiner Datenbank übernommen, um so meine Models zu erschaffen. Es gibt auch die Möglichkeit die Models zu autogenerieren, aber ich wollte, dass ich es etwas besser verstehe und auch weiss, wie es aufgebaut ist.

20.11.2022

Verwendete Ressourcen:

- Microsoft Dokumentation über SqlClient: https://learn.microsoft.com/en-us/sql/connect/ado-net/introduction-microsoft-data-sqlclient-namespace?view=sql-server-ver16

Da ich über eine Längere Zeit keinen Fortschritt mit dem Entity Framework gemacht habe, habe ich mich dazu entschieden das Entity Framework aus meinem Projekt zu löschen und es mit etwas anderem zu versuchen. Ich habe es versucht, indem ich mir ein Package namens System.Data.SqlClient heruntergeladen habe. Somit konnte ich SQL-Statements von Hand ausfüllen.

21.11.2022

Verwendete Ressourcen:

- Microsoft Dokumentation über Sessions: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-7.0

Problem:
Ich habe ein fertiges Login aber keine Session, welche gestartet wird, sobald man sich eingelogged hat.

Ich habe versucht über die Dokumentation von Microsoft eine Lösung zu dem Problem zu finden. Ich wusste wo der Start der Session sein musste, nämlich nach der Verifizierung des Logins, aber das Problem war, dass ich nach dem Einloggen, einen Redirect zur Homepage mache. Das Problem liegt darin, dass die Session nur auf den Seiten funktionieren, auf denen der Controller Zugriff hat. Da ich einen separaten Controller für mein Login und die Homepage habe, konnte ich keine Session starten, mit welcher ich auf der Homepage auf Session Variablen zugreifen kann. Bis jetzt konnte ich das Problem nicht lösen.

Problem:
Ich muss eine N zu M Beziehung auf der Website darstellen lassen.

Ich habe versucht eine Prozedur in der Datenbank zu schreiben, wo mir alle Datensätze von zwei Tabellen ausgegeben wird. Ich habe es mit einem Join versucht, aber schlussendlich habe ich es nicht geschafft, weil ich mehrere Muskelgruppen pro Übung erhalten habe und ich Mühe hatte dies Tabellarisch darzustellen.

	Exercise Name	Muscle Name
1	Bench Press	Back
2	Bench Press	Chest
3	Pull Up	Chest

22.11.2022

Verwendete Ressourcen:

- Kommentar erklärt Sessions und dass man Parent Controller erstellen kann: https://stackoverflow.com/questions/14138872/how-to-use-sessions-in-an-asp-net-mvc-4-application
- Wie man einen Redirect macht und einen Parameter einfügt: https://stackoverflow.com/questions/10785245/redirect-to-action-in-another-controller

Problem:
Session funktionieren immer noch nicht

Lösung:
Ich habe eine Lösung gefunden, mit welcher ich mein Problem mit den Sessions lösen kann. Die Lösung involviert einen Parent Controller, in welchem die Sessions festgehalten werden können. Die anderen Controller können auf den Parent Controller zugreifen und somit auch auf die Session Variablen.

24.11.2022

Ich habe im Unterricht erfahren, dass Sessions nicht nötig sind in einer ASP .NET Web-Applikation damit ein Login funktioniert. Ich kann anstatt Sessions, Claims und Cookies benutzen. Somit waren meine letzten paar Tage Arbeit unnötig.

25.11.2022

Verwendete Ressourcen:

- OneNote → M133 → Claims

Problem:
Ich habe immer noch kein Weg wie ich wissen kann ob ein Benutzer eingelogged ist oder nicht.

Lösung:
Da ich jetzt weiss, dass ich mit Claims und nicht mit Sessions arbeiten muss war es noch recht einfach, dass einzurichten. Ich musste nur nach dem erfolgreichen einloggen, die Claims und Claims Identity erstellen und danach ein SignInAsync machen.

```
await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
    new ClaimsPrincipal(claimsIdentity)); // Task
```

27.11.2022

Verwendete Ressourcen:

- Wie man eine ListBox macht: https://stackoverflow.com/questions/65556613/how-to-create-multiselect-dropdown-with-checkboxes-in-asp-net-mvc
- Wie man mehrere Checkboxes an den Controller weiter gibt: https://www.aspsnippets.com/Articles/ASPNet-Core-MVC-Pass-Send-CheckBox-values-to-Controller.aspx

Problem:
Muskelgruppen müssen beim Erstellen einer Übung angezeigt werden.

Ich habe verschiedene Varianten gesucht dieses Problem zu lösen. Zuerst wollte ich eine Dropdownliste erstellen, welche Checkboxes erhält. Nachdem ich ein wenig recherchiert habe, habe ich nur Lösungen gefunden, welche JavaScript und Bootstrap involviert haben. Da ich es so simple wie möglich halten wollte, wollte ich die Muskelgruppen mit einer ListBox aufzählen lassen. Eine ListBox hat alles jedoch nur noch komplizierter gemacht, da es kompliziert war, alle Muskelgruppen in eine ListBox zu bekommen. Ich habe es zwar geschafft, aber das UX war nicht sehr praktisch und könnte für viele Benutzer der Website unpraktisch sein.

Lösung:
Ich habe mich schlussendlich für normale Checkboxes entschieden. Ich hielt es ursprünglich nicht für möglich, da ich nicht wusste, dass mehrere Checkboxes direkt als String Array an den Controller zurückgegeben werden.

28.11.2022

Verwendete Ressourcen:

- Wie man einen Sha256 Hash in C# erstellt: https://www.techiedelight.com/generate-sha-256-hash-of-string-csharp/

Problem:
Benutzer können noch nicht registriert werden, da ich das gesamte System von auf Entity Framework gewechselt habe.

Lösung:
Ich habe das Passwort in C# und nicht in der Datenbank gehashed. Somit konnte ich den gesamten Insert Befehl mit dem Entity Framework machen und konnte einfach das gehashte Passwort in die Datenbank schreiben.

30.11.2022

Weil ich nicht unter den Workouts, alle Muskelgruppen aller Übungen ausgeben wollte, habe ich mich dazu entschieden, eine Details Page zu erstellen, auf welcher alle Details einer Übung zu sehen sind. Ich habe mich dazu entschieden, dass man von der Page der Workouts, wenn man auf eine Übung klickt, direkt auf die Details Page kommt.

01.12.2022

Verwendete Ressourcen:

- N zu M Abfrage mit C#: https://community.abp.io/posts/many-to-many-relationship-with-abp-and-ef-core-g7rm2qut

Heute im Unterricht habe ich gelernt, dass man die Namen der SQL-Tabellen nicht nach den Models richten muss, da man in den Models die Tabellennamen definieren kann. Deswegen konnte ich die Namenskonvention nochmals umändern.

Problem:
Ich versuche immer noch eine Lösung zu finden, um eine N zu M Beziehung auf der Website darstellen zu lassen.

Lösung:
Ich habe jeweils eine weitere Datenbankabfrage innerhalb eines Models gemacht, um so über die Hilfstabelle auf die nächste Tabelle zugreifen zu können.

```
return _databaseContext.Exercises.Select(e:Exercise => new Exercise
{
    Id = e.Id,
    Name = e.Name,
    Description = e.Description,
    ExerciseMuscles = _databaseContext.ExercisesMuscles.Where(em:ExerciseMuscle => em.ExerciseId == e.Id).Select(em:ExerciseMuscle =>
        new ExerciseMuscle
        {
            ExerciseId = em.ExerciseId,
            MuscleId = em.MuscleId,
            Muscle = _databaseContext.Muscles.First(m:Muscle => m.Id == em.MuscleId)
        }).ToList() // List<ExerciseMuscle>
}).ToList(); // List<Exercise>
```

03.12.2022

Problem:
Alle meine Datenbankabfragen waren synchron.

Lösung:
Es war sehr einfach alles auf asynchron umzustellen, da man nur eine Methode async machen musste, den Returntype zu einem Task<Returntype> machen und vor dem Abfragestatement noch await schreiben.

06.12.2022

Verwendete Ressourcen:

- Wie man ThenInclude macht: https://stackoverflow.com/questions/13047845/how-to-include-a-child-objects-child-object-in-entity-framework-5#answer-43075537

Problem:
Die Art und Weise wie ich die N zu M Verbindung auf der Website darstellen lasse, ist sehr umständlich.

Lösung:
Ich habe es mit Include versucht, weil man da aus mehreren Tabellen abfragen kann. Das Problem war, dass ich nicht mehrmals einen Include machen konnte, welcher auf das Model vom letzten Include zugriff hatte. Als ich aber TheInclude gefunden habe, war das Problem sehr einfach zu lösen.

```
return await _databaseContext.Posts // DbSet<Post>
    .OrderByDescending(p :Post => p.Date)
    .ThenByDescending(p :Post => p.Id) // IOrderedQueryable<Post>
    .Include( navigationPropertyPath: p :Post => p.User) // IIncludableQueryable<Post,User>
    .Include( navigationPropertyPath: p :Post => p.PostWorkout) // IIncludableQueryable<Post,ICollection<...>>
    .ThenInclude(pw :PostWorkout => pw.Workout) // IIncludableQueryable<Post,Workout>
    .ThenInclude(w :Workout => w.WorkoutExercises) // IIncludableQueryable<Post,ICollection<...>>
    .ThenInclude(we :WorkoutExercise => we.Exercise) // IIncludableQueryable<Post,Exercise>
    .ToListAsync(); // Task<List<...>>
```

07.12.2022

Problem:
Bei einer N zu M Beziehung werden die Daten, welche in die Hilfstabellen eingefügt werden müssen, nicht bearbeitet.

Lösung:
Nachdem ich es geschafft habe, einen Insert Befehl einer N zu M Beziehung zu machen, habe ich das Prinzip hinter dem Bearbeiten auch verstanden. Ich habe bisher versucht die Foreign Keys von Hand in die Datenbank einzufügen, indem ich einen Insert Befehl auf die Hilfstabelle gemacht habe. Ich musste jedoch nur dem ersten Model, welche die Hilfstabelle in sich hat, das Model der Hilfstabelle zuteilen.

Problem:
Es können keine Datensätze in den Tabellen Workout und Exercise von der Website aus gelöscht werden.

Lösung:
Ich habe mir zuerst Sorgen gemacht, weil es beide N zu M Beziehungen sind, aber das Problem war sehr einfach zu lösen, da man nur das Model mit der Id auswählen musste und danach konnte man mit dem Model konnte man direkt etwas aus der Datenbank löschen.

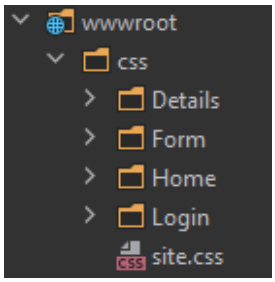
08.12.2022

Verwendete Ressourcen:

- Unterschied zwischen RedirectResult und RedirectToActionResult: https://stackoverflow.com/questions/12198909/what-is-the-difference-between-redirect-and-redirecttoaction-in-asp-net-mvc

Problem:
Ich habe eine einzige CSS-Datei namens site.css, welches sehr unübersichtlich ist.

Lösung:
Ich teile meine Datei in mehrere Dateien auf. Dafür musste ich nur eine klare Ordnerstruktur erstellen. Danach musste ich noch die verschiedenen Files in den richtigen View Dateien verlinken.



Problem:
Bei einem Redirect, wird die URL angegeben um den Benutzer weiterzuleiten.

Lösung:
Ich habe den Returntype von RedirectResult zu RedirectToActionResult geändert. RedirectToActionResult hat nämlich den Unterschied, dass man der Name einer Methode, des Controllers und wen nötig sogar einen Wert im Parameter angeben kann, anstadt der Name einer URL. Bei weiterer Recherche habe ich herausgefunden, dass es am meisten sinn macht, um RedirectToActionResult zu verwenden, wenn es um die eigene Applikation geht und RedirectResult wenn es um Links ausserhalb der Applikation geht.

10.12.2022

Problem:
Man muss auf der Website sehen können ob ein Benutzer einen Admin ist oder nicht.

Lösung:
Im Model des Benutzers habe ich einfach ein neues Attribut hinzugefügt namens Admin. Der Admin

hat den Datentyp bool. Da man in dem gesamten Programm aber niemals der Wert vom Admin neu setzten kann, habe ich nur einen Getter ohne einen Setter gemacht. Als ich, dass gemacht habe, habe ich nur False als Rückgabewert erhalten, auch wenn der Benutzer in der Datenbank ein Admin ist. Die Lösung war es, dem Attribut einen Setter zu geben. Danach erhielt ich immer den richtigen Rückgabewert aus der Datenbank und ich konnte wissen wer ein Admin ist und wer nicht.

Projekt Fazit

Mit C# eine Web-Applikation zu erstellen war eine komplett neue Erfahrung, die ich mit keinem anderen Projekt gemacht habe. Ich habe bisher mit PHP nach MVC eine Web-Applikation gemacht aber mit C# hat das um einiges mehr Spass gemacht. Was mir unerwartet am meisten Zeit gekostet hat, war es die gesamten Informationen herauszusuchen, welche ich für das Projekt benötigte. Als ich die Informationen zusammen hatte, konnte ich sehr schnell die Grundbausteine für die Web-Applikation erstellen.

Ich bin sehr froh darüber, dass ich das Projekt mit Rider anstatt Visual Studio gemacht habe. Rider hat mir keinerlei Probleme gemacht und hatte ein paar gute Features.

Wenn ich am Ende noch mehr Zeit gehabt hätte, wären wahrscheinlich noch weitere Features dazu gekommen wie zum Beispiel:

- Upvote / Downvote System bei den Posts
- Workouts können von den Posts zu den eigenen kopiert werden
- Account Verwaltung (Benutzername / Passwort ändern)
- Admins können Benutzer sperren