Modul 151

Thema: Quizleter

Dokumentinformationen

Dateiname: Dokumentation\_Quizleter

Speicherdatum: 01.11.2022

Autoreninformationen

Autor: Venturini Alessio, Ronnie Hartmann

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc118204603)

[1.1 Beschreibung des Projektes 3](#_Toc118204604)

[2 Architektur 4](#_Toc118204605)

[2.1 MVC (Model View Controller) 4](#_Toc118204606)

[3 Frameworks 4](#_Toc118204607)

[4 ERM 5](#_Toc118204608)

[5 Testing 6](#_Toc118204609)

[5.1 Unit Tests 6](#_Toc118204610)

[6 Fazit & Reflexion 7](#_Toc118204611)

# Einleitung

## Beschreibung des Projektes

Projekt hat den Namen «Quizleter» und ist ähnlich aufgebaut wie die App Quizlet. Wenn man die Seite betrete, sieht man eine einfache Startseite, welche aber jedoch noch nicht viel aussagt.

Wenn man die App benutzen will, muss man sich zuerst auf unserer Seite anmelden, damit man die Funktionen auch freischaltet. Wenn man noch keinen Account bei uns besitz, kann man sich einen Account erstellen. Sobald man sich korrekt einen Account erstellt hat, kann man sich mit seinen Daten anmelden. Nach der Anmeldung kann man die verschiedenen Funktionen verwenden.

Auf der «Learnsets» Seite sieht man alle Lernsets, die gerade existieren. Man kann auch nach einem bestimmten Lernset suchen, wenn man es nicht finden kann. Wenn einem kein Lernset gefällt, kann man sich auch ein eigenes Lernset erstellen. Bei der Erstellung muss man nur die Wörter angeben und ist fertig. Nun sollte auch das Lernset angezeigt werden.

Wenn man sich mit einem Lernset befassen möchte, kann man auf das Lernset klicken. Danach werden dem Benutzer einige Funktionen vorgeschlagen, die der Benutzer verwenden kann, um die Wörter zu lernen.

Wenn der Benutzer sich für den Karteikarte-Modus entscheidet, lernt er die Wörter mit der Hilfe von Karteikarten. Bei diesem Modus werden die Wörter als Karteikarten dargestellt. Wenn man die Übersetzung sehen möchte, kann man über die Karteikarte fahren und man sieht die andere Seite.

Der zweite Modus, den man wählen kann, ist der Lernmodus. Man lernt die Wörter, indem man sie schreibt. Jedes Wort hat eine Skilllevel, welches beschreibt, wie gut der Benutzer das Wort schon kann. Wenn man ein Wort richtig angibt, wird das Skilllevel um eins erhöht. Wenn man das Wort falsch angibt, wird das Skilllevel um eins verkleinert. Jederzeit kann der Lernende eine Auswertung einfordern, welche dem Benutzer zeigt, welches Wort er schon am besten kann. Sobald er bei jedem Wort ein Skilllevel von zehn erreicht hat, wird er automatisch auf die Auswertung geleitet, weil er die Wörter beherrscht. Wenn er jedoch nochmals die Wörter lernen möchte, kann er die Skills zurücksetzen und die Wörter nochmals lernen.

Der letzte Modus ist ein Test-Modus. Dieser Modus ist recht ähnlich wie der Lernmodus, nur in einer Testform. Der Benutzer schreibt einen kurzen Test und am Ende sieht er, welche Wörter er richtig hat und die erreichte Punktzahl.

# Architektur

## MVC (Model View Controller)

Wenn man das Backend und das Frontend zusammen in einem Projekt entwickeln möchte, eignet sich das MVC Pattern. Wie es der Name schon sagt, besteht ein solches Projekt vor Allem aus Models, Views und Controller.

### Model

Ein Model ist im Grunde genommen eine einfache Klasse mit verschiedenen Feldern. Mit .NET kann man dann aus den Models Tabellen in einer Datenbank erstellen. Das nennt sich Code-First. Das geht jedoch auch umgekehrt. Wenn man bereits Tabellen hat, aber den Code noch schreiben muss, kann man die Models generieren lassen (Database-First).

public class Person

{

public long Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime BirthDate { get; set; }

}

Das wäre zum Beispiel ein Model für eine Person. Der Code ist in C# geschrieben.

### View

Die View ist das, was der Benutzer sieht. Eine View besteht oft aus HTML. Oft nimmt eine View ein Model entgegen, um die Daten des Models anzuzeigen. Noch besser ist es jedoch, ein DTO an die View zu geben.

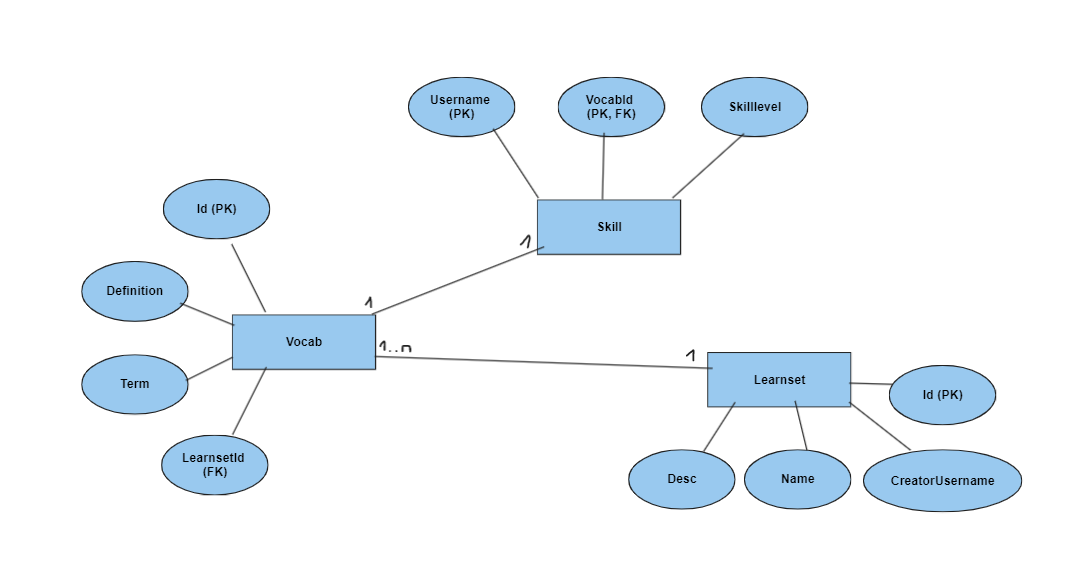
### Controller

In den Controllern werden alle Endpoints definiert. Es mach Sinn, für jedes Model einen eigenen Controller zu erstellen. Der Controller autorisiert den Benutzer und verarbeitet und validiert die Daten. Oft verwendet man noch Services, um die Logik vom Controller auszulagern. Die Services werden dann mit Hilfe von Dependency Injection in den Controllern angewendet.

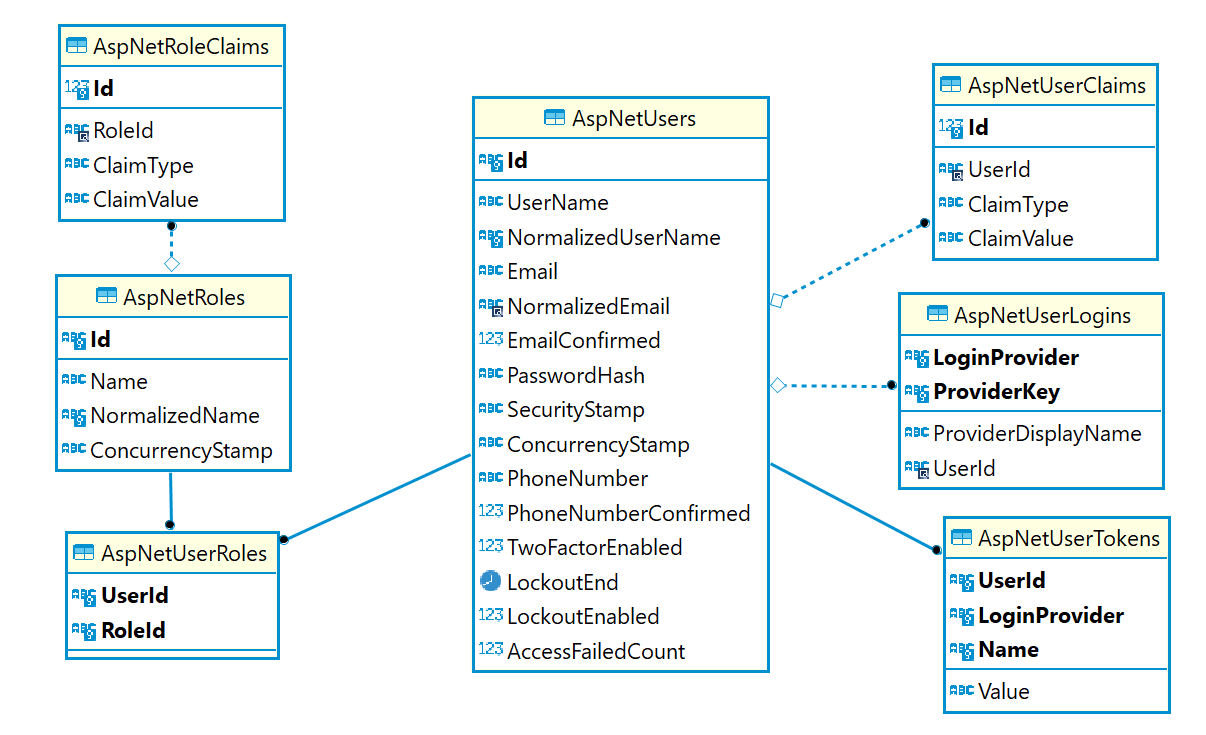
# Frameworks

* EF Core (Entity Framework)
* Microsoft Identity
* .Net 5 Framework

# ERM



Wir haben zwei verschiedene Datenbanken. In der oben gezeigten Datenbank werden die Lernsets, Wörter und der dazugehörige Skill gespeichert. Die Tabelle «Learnset» beinhaltet noch Informationen von der zweiten Datenbank, welche sich um das Log-in und alles, was dazugehört kümmert. Ein Lernset beinhaltet mehrere Wörter. Jedes Wort hat einen Begriff und eine Definition. Ebenfalls hat jedes Wort einen dazugehörigen Skill. Der Primärschlüssel dieser Tabelle ist aus dem Usernamen des aktuellen Benutzers und aus der Id des Wortes zusammengesetzt. Die Id des zusammengesetzten Wortes ist ebenfalls ein Fremdschlüssel.



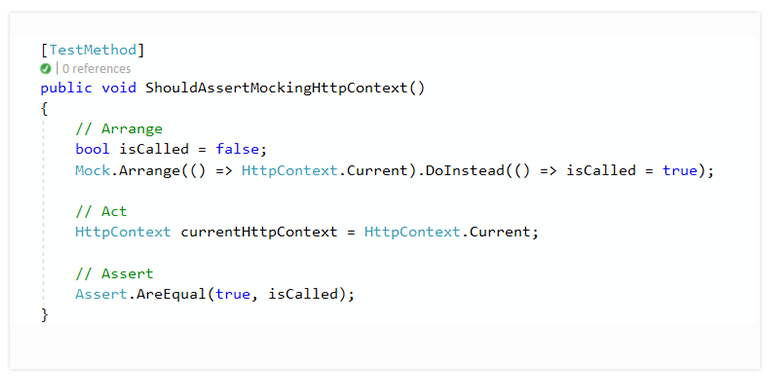
# Testing

## Unit Tests

In Unit Tests sind Tests, die in der Applikation geschrieben werden. Wenn man ein Projekt testen möchte, sollte man ein zweites Projekt in derselben Solution erstellen, somit kann man die Tests von dem Projekt trennen und besser verwalten.

Bei Unit-Tests ist es wichtig, dass man nicht alles testet, sondern sich nur auf wirklich einen Teil bezieht. Wenn in der Methode sich eine weitere Methode befindet, testet man diese nicht auch noch dazu, sondern bezieht sich nur auf die aktuellen Methoden. Ein weiterer wichtiger Punkt ist, wenn man eine Methode testet, die Daten aus der Datenbank zurückgibt, nicht mit den Daten aus der Datenbank arbeitet, sondern mit Testdaten. In .Net kann man einen Moq von dem Kontext erstellen und danach Testdaten zuweisen.

### Beispiel



Quelle: <https://www.telerik.com/products/mocking/utilize-arrange-act-assert-aaa-pattern.aspx>

### Arrange

In den Unit-Tests arbeite ich mit den Arrange/Act/Assert Aufteilung. Mit dieser Strategie kann man seine Unit-Tests gut aufteilen. In dem Arrange Teil kommen die Initialisierung oder Testdaten, die man holt.

### Act

Im Act wird die Methode aufgerufen und die Testdaten in die Methode gegeben. Danach werden die Daten in die Testmethode geholt, damit die Daten getestet werden können.

### Assert

Im Assert werden die Daten getestet.

# Fazit & Reflexion

Da wir im Geschäft oft mit MVC arbeiten, konnte wir uns gut auf diese Architektur einigen. Bei der Arbeit am Projekt hatten wir keinen Zeitdruck und hatten am letzten Tag genügend Zeit für die Dokumentation. Die Arbeit an der Lern-App machte uns Spass, da uns C# und .NET sehr gefällt und die Applikation auch später hilfreich für uns sein kann. Vielleicht werden wir die Website in Zukunft veröffentlichen, damit unsere Klasse sie für Voci-Prüfungen verwenden kann.