# Dokumentation zur Anwendung für Programmierer:

Stand: 02.03.2020

## Einleitung zur Technologien

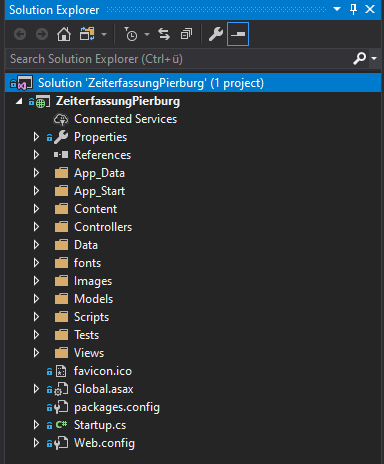
Die Anwendung wurde in Visual Studio 2017 entwickelt mit MVC Design Muster.   
  
Es wurde dabei die Vorlage von ASP.NET MVC Framework angewendet. Als Datenbankserver wurde der SQL Server 2017 benutzt.  
Für die Lösung wurde nicht das Entity Framework benutzt, was oft der Standard ist, sondern für das Mapping der Datenbank Queries und des Models wurde das Package Dappereingesetzt. Das heißt, viele Funktionen wurden an den Datenbankserver verlegt, um mehr Flexibilität bei Änderung der Datenbank zu gewinnen.

Dokumentation und Tutorial zur MVC: <https://dotnet.microsoft.com/apps/aspnet/mvc>

Dokumentation und Tutorial zum Dapper: <https://dapper-tutorial.net/>

Für das Frontend wurde kein Framework angesetzt, sondern einfaches Javascript, ergänzt mit Bootstrap und teilweise JQuery.

## Überblick

****

Die Hauptfunktionalitäten sind In Models, Controllers und Views Foldern. Die weitere wichtige Klasse ist der SQLServer.cs in Data Folder.

## Datenbank

Die Datei zur Herstellung der Datenbank befindet sich unter SQL\_Scripts.sql in ~\ZeiterfassungPierburg\ZeiterfassungPierburg\bin\Release\Publish.

Es gibt 5 Tabellen, die miteinander mit Fremdschlüssel verbunden sind.   
Die Kerntabelle ist dabei die MitarbeiteInSchicht, wo die gesamten Informationen über einzelne Schichteinträge enthalten sind.  
Die Zugriffsrechte Tabelle dient der Speicherung der Benutzerdaten und TeileInProduktionsanlage verkuppelt die Produktionsanlage und Fertigungsteil Tabelle, jedoch ohne Fremdschlüssel, ohne unnötige Fehler zu verursachen.

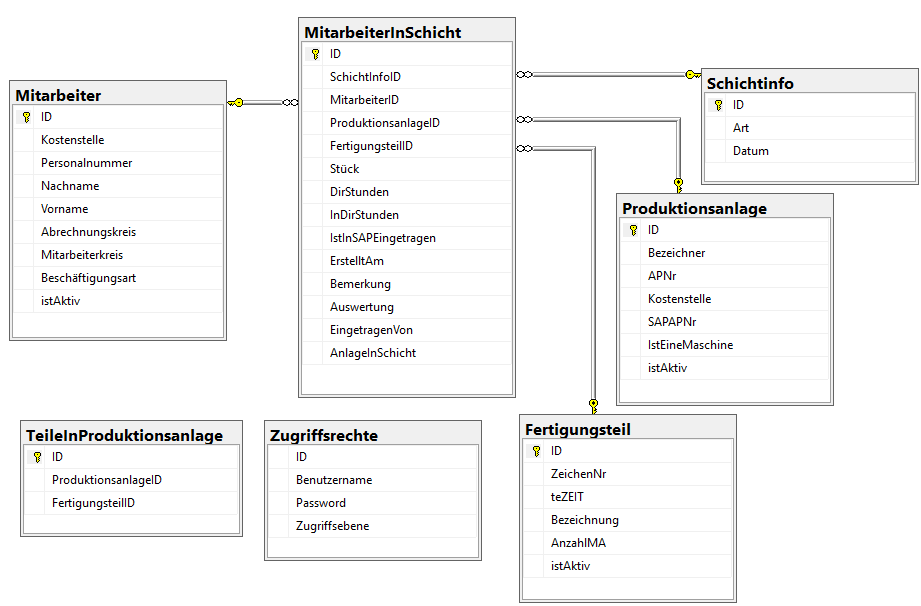


Abbildung 1: Datenbank Schema

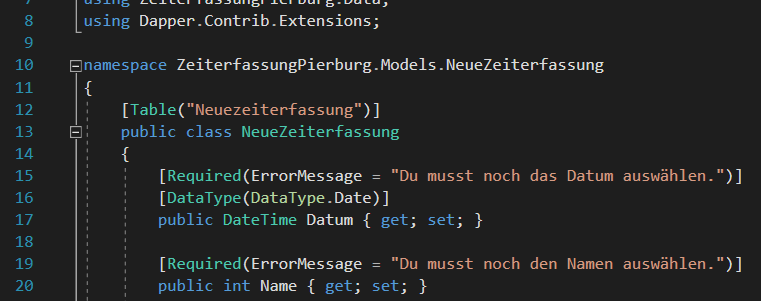
Jeder Eintrag entspricht immer einer ID in MitarbeiterInSchicht Tabelle. Schichtinfo enthält Informationen über den Zeitpunkt (bei Art 1= Frühschicht, 2=Spätschicht, 3=Nachtschicht). Produktionsanlage enthält Info über Bänder und Maschinen, wobei der einzige Unterschied zwischen den beiden ist der bool Wert *IstEineMaschine*.

## Models

In Models Folder sind die Modelklassen und auch die Viewmodels gespeichert. Es wird nicht deutlich zwischen den beiden unterscheidet, die meisten Modelle dienen gleichzeitig auch als Viewmodels.

Alle Modelle erweitern die *BasicModelObject* Klasse. Diese Klasse enthält die ID des Models, wessen Key für Dapper notwendig ist.

## Mapping

Das Mapping von Modellen und den Datenmodellen erfolgt mithilfe der Dapper Package . Das Mappen selbst passiert in Controller resp. In SQLServer Klasse; was man jedoch beachten muss ist, dass die Tabelle in SQL Server genauso wie die Klassen, sonst ist es notwendig, den Tabellenamen als Attribut hinzuzufügen. Die Tabellennamen (oder Namen der Spalten in einer SQL Query) und die Properties des Models müssen auch genauso heißen, damit das Mapping erfolgreich ist. 

## SQLServer.cs

SQL Server Klasse ist der Kern der Anwendung. Einzelnen Methoden ziehen die Daten aus dem Server und arbeiten weiter mit Ihnen für die Nutzung der Anwendung.

Die Klasse ist sowohl für das Erstellen eines Modells als auch für Berechnungen anderer Daten wie z.B. der Produktivität. Wenn man in VS alle Methoden einklappt, sieht man die Beschreibung der Gruppen der Funktionen.

Die wichtigsten Methoden sind die Datenmanipulierungsmethoden (CRUD).

Künftige Verbesserung wäre z.B. eine Einteilung in mehreren Klassen von wegen Übersichtlichkeit.

### Berechnung der Produktivität

## Controllers

## Views

Als Designvorlage wird als Layout AdminLTE <https://github.com/ColorlibHQ/AdminLTE> verwendet. Es ist eine css und js Erweiterung von Bootstrap.   
Die meisten Views sind direkt an die Models gebunden und sind nur leicht unterschiedlich von den defaulten CRUD Views generiert von ASP.NET.

Die nicht-Standarte Views sind die in folgen Folders: *Home, Neuezeiterfassung, TeileInProduktionsanlage, Mitarbeiter/Details, Produktivität*

Für alle anderen gilt folgend:

*Create* und *Update* werden direkt von den Modellen abgeleitet. Die Validierung kommt auch von den Modellen und kein zusätzliches JS Code ist geschrieben.  *Index* Views sind von Bootstrap‘s DataTables erweitert:  
<https://datatables.net/examples/styling/bootstrap4>

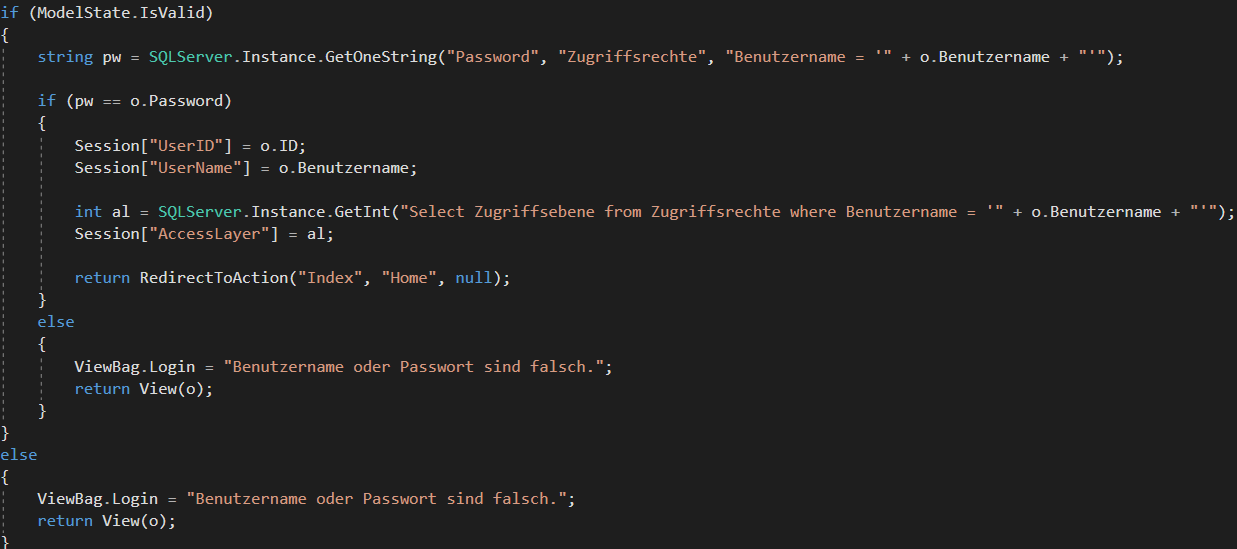
Es ist eine einfache css und Javascript Erweiterung, die vereinfacht die Ansicht der Tabelle und ermöglicht die Sortierung und Suche.

### NeueZeiterfassung

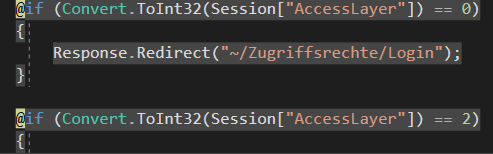
### 

## Login und Benutzerverwaltung

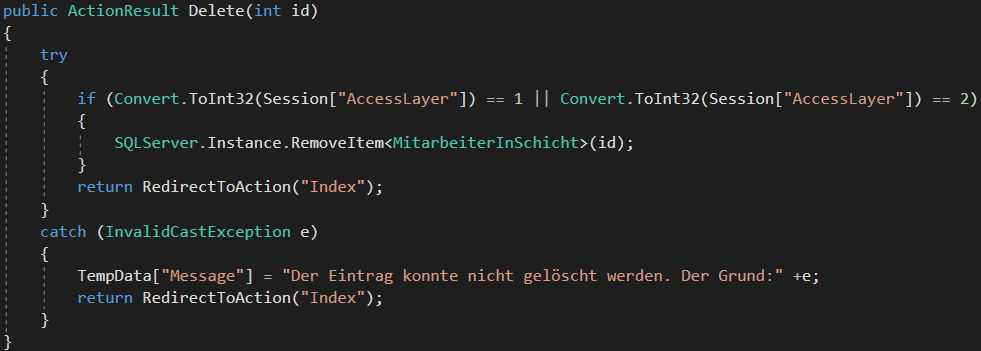
Es war gewollt, dass man keine Windows Authentifizierung benutzen sollte (?), deswegen ist in der Anwendung noch eine primitive Sicherheitsschichte mit Benutzernamen und Passwort implementiert. Die Passwörter sind nicht gehashed und werden in Klartext in der Datenbank gespeichert.

Beim Einloggen wird tatsächlich nur der Benutzername mit dem Password verglichen. Im Controller wird beim Hinzufügen eines Benutzernamens gesorgt, dass dieser eindeutig ist.  


Es gibt 2(3) Unterschiedlich Zugriffschichten: der Admin, der Verwalter (und der Benutzer ohne Login).   
Der uneingeloggte Benutzer hat nur Zugriff auf „Neue Zeiterfassung“. Der Verwalter auf alles andere außer Zugriffsrechte. Der Admin darf auch Zugriffsrechte vergeben.   
Der Zugriff wird nur damit eingeschränkt, in dem es 2 verschiedene Layouts gibt. Einmal ist das der Layout für öffentlichen Zugang *\_LayoutPublic* und für die eingeloggten User der *\_Layout*. Im *\_Layout* wird der AccessLayer befragt:



Je nach dem AccessLayer wird unterschiedlicher Inhalt geladen. Soweit ist der Unterschied nur das Menuitem *Zugriffsrechte*. In Views, die unter *Zugriffsrechte* gehören, wird der AccesLayer von SessionInfo nochmal gefragt und wenn der Benutzer kein Admin ist, wird es oder sie auf AccessDenied Seite umgeleitet.  
Da *Delete* Methoden nur über dir URL erfolgen, wird in jeweiligen Controllers auch nach der Zugriffsschichte gefragt, wie am Beispiel von *MitarbeiterInSchichtController*.



Die von ASP.NET generiert MVC Inhalte sind für eventuelle spätere Implementierung beibehalten, in dieser Lösung werden jedoch nicht benutzt.