Projektlaborbericht

Mit dem Thema:

Programm für die Zeiterfassung von Kleinunternehmen

eingereicht durch: Name: David Kempf, Sean Woods, Lauritz Abel

Matrikelnummer: 207815, 207820, 207804

Studiengang: AE

Hochschule Heilbronn – Campus Künzelsau

betreut von: Prof. Dr.-Ing. Marcus Stolz

Hochschule Heilbronn – Campus Künzelsau

Künzelsau, xx.xx.xxxx

Ehrenwörtliche Erklärung

Hiermit erkläre wir, David Kempf, Sean Woods und Lauritz Abel dass wir die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Es wurden keine anderen als die angegebenen Quellen und Hinweise verwendet.

Die vorliegende Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Künzelsau xx.xx.xxxx David Kempf, Sean Woods, Lauritz Abel

Inhaltsverzeichnis

[1. Ist-Zustand: 1](#_Toc140042146)

[2. Kurzfassung 1](#_Toc140042147)

[3. Aktuell Verfügbare Systeme 2](#_Toc140042148)

[TOGGL 2](#_Toc140042149)

[AtWork 2](#_Toc140042150)

[Factro 2](#_Toc140042151)

[4. Entwicklung 3](#_Toc140042152)

[GitHub 3](#_Toc140042153)

[JetBrains Rider 3](#_Toc140042154)

[.NET-Framework 4](#_Toc140042155)

[WPF .NET 4](#_Toc140042156)

[C# 5](#_Toc140042157)

[Fazit und Ausblick 6](#_Toc140042158)

[1. Abbildungsverzeichnis 7](#_Toc140042159)

# Ist-Zustand:

Für kleine Handwerks- und Landschaftspflegebetriebe besteht aktuell das Problem, es gibt keine mobile digitale Lösung für die Zeitdokumentation. Momentan dokumentieren viele kleine Betriebe ihre Zeiten in kleinen Notizbüchern. Maschinenstunden werden wegen des Aufwandes oft nicht richtig dokumentiert und die Zeiten geschätzt. Es gibt die Möglichkeit mittels Produkte wie zum Beispiel von Stihl (Stihl Connector) Maschinenzeiten aufzunehmen und zu dokumentieren. Jedoch müssen diese wieder händisch in ein Rechnungstool übertragen werden. Ein weiteres Problem ist, dass die Büroarbeit von Handwerker oft als lästig und unnötig betrachtet wird und somit wieder zu kurz kommt. Jedoch verdient ein Betrieb sein Geld nicht nur mit der verrichteten Arbeit, sondern mit dem Ausstellen von korrekten Rechnungen.

# Kurzfassung

Um den aktuellen Zustand zu verbessern ist das Ziel dieser Arbeit, mit Hilfe einer App die Zeiterfassung und Rechnungsstellung zu automatisieren. Das Programm soll in zwei Bereiche aufgeteilt werden. Eine Desktopanwendung, welche zur Aufgabenplanung und zum Controlling genutzt werden soll und eine mobile App welche als Anwenderapp, für die Zeiterfassung gedacht ist. Das Programm soll es dem Anwender ermöglichen leicht seine Arbeitszeiten und Maschinenzeiten zu dokumentieren und zu Verwalten. Nachdem ein Auftrag abgearbeitet ist, soll es mittels eines Klicks möglich sein, eine Rechnung erstellen zu lassen. Somit wird die Zeit im Büro minimiert und die effektive Arbeitszeit erhöht.

# Aktuell Verfügbare Systeme

## TOGGL

Toggl ist ein großes Zeitverwaltungssystem welches von Unternehmen wie Amazon, SAP, Linkedin usw. genutzt wird. Das Programm besticht mit einer einfachen Bedienung, somit kann auch ohne eine Schulung gearbeitet werden. Die Software ist sehr offen aufgebaut und lässt sich mit andere Softwaren wie Jira, Salesforce usw. verbinden.

## AtWork

AtWork ist eine kostenpflichtiges Zeiterfassungssystem. In diesem lässt sich das ganze Mitarbeiterteam verwalten. Das Programm sticht vor allem mit der mit der Funktion der Überstundenberechnung und Auszahlung und der Funktion mit der Schichtplanug heraus

## Factro

Fracto ist eine Cloud basierte App. Diese speichert die Arbeitsstunden Projekt bezogen ab. In der App ist es möglich ein ganzes Team zu verwalten. Mithilfe der Controlling Funktion ist es möglich die Auslastung des Teams auszuwerten und zu bewerten. Eine kleine Basis-Version ist kostenlos danach müssen kostenpflichtige Lizenzen gekauft werden.

Die bestehenden Anwendungen sind oft sehr teuer und somit nicht für kleine Betriebe rentabel oder geeignet. Die kostenlosen Apps sind oft unflexibel und eigenen sich nicht für den kommerziellen Gebrauch. Momentan fehlt auf dem Markt eine einfache und flexible Anwendung, welche sich auf die relevanten Funktionen begrenzt.

# Entwicklung

## GitHub

Was ist GitHub? GitHub ist eine Cloudbasiert Software, welche die Datenverwaltung und den Projektablauf vereinfacht. Somit ist es möglich das mehrere Entwickler gleichzeitig an einem Sourcecode schreiben können, oder Daten gemeinsam geteilt werden. Die neu entwickelten Inhalte können mit allen geteilt werden und werden versioniert abgespeichert. Somit ist für alle ersichtlich, was geändert oder hinzugefügt wurde. Nicht nur Sourcecode ist möglich, sondern auch andere Dateien, wie Word, Excel, PowerPoint und vieles mehr.

## JetBrains Rider

JetBrains Rider oder kurz nur Rider genannt, wurde als Entwicklungsumgebung gewählt, da dieser einige Vorteile mit sich bringt. Rider ähnelt sehr stark der Entwicklungsumgebung Visual Studios von Microsoft. Hier liegt auch schon ein großer Unterschied, Rider ist nicht von Microsoft. Rider ist eine komplett offene Entwicklungsumgebung und nicht an ein Betriebssystem gebunden. Außerdem ist die Entwicklungsumgebung sehr Benutzerfreundlich da diese eine XAML-Vorschau bietet, welche in Echtzeit die Xamarin XAML Seite zeigt. Bei einem Fehler im Sourcecode kann mithilfe der Codeanalyse der Fehler schnell ausgemacht und behoben werden. Dies wird durch den Rider unterstützt, da dieser sofort einen Lösungsvorschlag mitliefert. Tritt beim Ausführen ein Fehler auf unterstützt der Debugger den Entwickler. Dieser zeigt den Breakpunkt auf und den Grund, warum das Programm nicht ausgeführt werden konnte.

## .NET-Framework

Das .NET-Framework wurde von Microsoft entwickelt und bedeutet übersetzt so viel wie „Rahmenarbeit“. Hierbei dient das .Net Framework als Basisprogramm für die Programmierung von desktop- und mobilen Anwendungen. Es besteht aus einer Laufzeitumgebung, Klassenbibliothek und einem Compiler. Der geschriebene Sourcecode, welcher in der .Net Framework programmiert worden ist, wird in der Laufzeitumgebung ausgeführt. Die Klassenbibliothek dient als nützliche Quelle für jegliche Anwendungen, wie zum Beispiel für SQ Lite Datenbanken, PDF-Generatoren, Socket Server und so weiter. Um das .Net Framework nutzen zu können, wird ein Texteditor und ein Compiler benötigt. Hier bietet Microsoft ihr Visual Studios an, oder wie in diesem Fall den verwendeten JetBrains Rider.

## WPF .NET

Windows Presentation Foundation (WPF) ist eine Klassebibliothek von .NET, welche zu der Entwicklung von graphischen Oberflächen genutzt wird. WPF unterstützt verschiedenste Arten von GUIs: Typische Desktopanwendungen, 3D Grafiken, Dokumente, Browser-basierte Anwendungen und Videos. WPF ist von Microsoft als Nachfolger für das aus .NET bekannte Windows Forms eingeführt worden. Eine WPF-Oberfläche kann als Programmcode programmiert, oder wie von Rider unterstütz durch XAML-Language programmiert werden. Die Anwendung ist vektorbasiert aufgebaut. Daher ist die Auflösung unabhängig von der Bildschirmgröße. Diesen Vorteil wird richtig deutlich beim Verwenden eines responsiven Designs. Bei einem responsiven Design passt sich die grafische Oberfläche der Gesamtgröße an. Zieht man die Anwendung nun größer leidet die Auflösung nicht darunter.

## C#

C# wird in diesem Projekt als Programmiersprache verwendet. Diese ist eine objektorientierte Sprache. Ihren Ursprung hat diese auch bei Microsoft Anfang der 2000er Jahre. Die Sprache wurde Plattform unabhängig entwickelt im Rahmen der .NET-Strategie von Microsoft. Diese sollte das Erstellen von Windowsanwendungen mit der .NET-Framework vereinfachen. Hierbei greift die Sprache Konzepte von [Java](https://de.wikipedia.org/wiki/Java_(Programmiersprache)), [C++](https://de.wikipedia.org/wiki/C%2B%2B), [Haskell](https://de.wikipedia.org/wiki/Haskell_(Programmiersprache)), [C](https://de.wikipedia.org/wiki/C_(Programmiersprache)) sowie von [Delphi](https://de.wikipedia.org/wiki/Embarcadero_Delphi) auf.

Zeiger welche in C++ als eher unsicher bekannt sind werden in C# nur für den sogenannten „unsicheren Code“ zugelassen zum Beispiel Programme welche von einer Website ausgeführt werden ohne erweiterte Rechte. Da C# als .NET-Sprache gilt, gibt es auch die Möglichkeit über eine Sprachunterstützung für Attribute und Delegaten. In Metadaten werden in C# Informationen über eine Klasse, ein Objekt oder eine Methode gespeichert. Diese werden zur Laufzeit ausgewertet.

Aufbau Datenbank

Für die Datenspeicherung und Datenverwaltung wurde eine SQ-Lite Datenbank verwendet. SQ-Lite bietet die Möglichkeit direkt in die Anwendung eine Datenbank zu implementieren, ohne dass eine weitere Server-Software benötigt wird.  
Bei dem Erstellender Datenbank ist es wichtig keinen absoluten Pfad zu verwenden. Ein Indirekter Pfad hat den Vorteil, dass dieser, egal auf welchem PC die Anwendung ausgeführt wird, funktioniert. Ein absoluter Pfad birgt die Gefahr in sich, dass der Pfad nicht existiert und die Anwendung nicht ausgeführt werden kann. Hierbei empfiehlt sich der Debug Ordner zu verwenden, denn dieser ist unabhängig vom Installationsort.   
Ein Bild, das Screenshot, Text, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung  
Außerdem muss die Funktion „All schemas“ aktiviert werden. Mittels dieser Funktion werden alle später erstellten Tabellen angezeigt. Für jede Datenbank wird in dem Ordner DB einen Klasse erstellt. Diese wird mit einem Rechtsklick und dem Befehl „Add“ hinzugefügt. In der Public Class wie zum Beispiel „public class Db\_Customer“ werden sämtliche „Eigenschaften“ für die Tabellenerstellung definiert. Hierfür beschreibt jede Zeile eine Spalte der Tabelle. In jeder Zeile muss der Spaltennamen und der Datentyp definiert sein.

public string Character { get; set; }  
public string BgColor { get; set; }  
public string ID { get; set; }  
public string Name { get; set; }  
public string Adress { get; set; }  
public string Mail { get; set; }  
public string Phone { get; set; }

Mit diesen Informationen kann nun eine Tabelle in der Datenbank erstellt werden. Hierfür wird eine Methode erstellt. Diese funktioniert nach dem try catch Prinzip. Dieses versucht die Funktion welche im try steht auszuführen, solange die Bedingungen für try erfüllt sind. Sind diese nicht erfüllt gehen diese automatisch in catch und führen die Funktionen in catch aus. In folgendem Sourcecodeabschnitt wird im try eine neue Tabelle erstellt solange diese noch nicht vorhanden ist. Falls eine vorhanden ist, wird eine Fehlermeldung ausgegeben.

public static Error CreateTable(string dataSource)  
{  
 SQLiteConnection conn = null;  
  
 try  
 {  
 conn = new SQLiteConnection(dataSource);  
 conn.CreateTable<Db\_Customer>();  
  
 return null;  
 }  
 catch (Exception ex)  
 {  
 return new Error(ex.ToString());  
 }  
 finally  
 {  
 conn?.Close();  
 }  
   
}

In dem nächsten Schritt muss nun die gerade erstellte Tabelle noch mit Zeilen und Inhalt gefüllt werden. Hierfür wird eine neue Klasse für das Schreiben und Lesen der Tabelle erstellt, zum Beispiel: „RW\_Customer.cs“.

*/// Reihen in DB schreiben  
/// </summary>*public static Error Write(List<Db\_Customer> rows, string dataSource)  
{  
 SQLiteConnection conn = null;  
  
 try  
 {  
 *//Aufbau Verbindung zur Datenbank* conn = new SQLiteConnection(dataSource);  
 *//Alle Reihen durchsuchen ob ID bereits vorhanden* foreach (var row in rows)  
 {   
 string where = $"WHERE ID='{row.ID}'";  
  
 *// pruefen ob Item/Reihe vorhanden ist* var item = conn.Query<Db\_Customer>($"SELECT \* FROM {nameof(Db\_Customer)} {where}*"*).FirstOrDefault();  
 if (item != null)  
 {  
 *// schon vorhanden, loeschen* conn.Execute($"DELETE FROM {nameof(Db\_Customer)} {where}*"*);  
 }  
 *//neue Reihe einfügen* conn.Insert(row);  
 }

Einfügen oder Beschreiben einer Zeile folgt immer die gleiche Methode. Zuerst wird eine Verbindung zu der Datenbank aufgebaut. Danach wird die jeweilige Tabelle durchsucht, ob die dementsprechende Reihe schon vorhanden ist und ob in dieser der zu ändernde Wert vorhanden ist. Ist dieser vorhanden wird die ganze Reihe durch den Befehl „conn.Execute($"DELETE FROM {nameof(Db\_Customer)} {where}");“ gelöscht. Der Befehl beinhaltet folgende Informationen: es sollen Daten aus der Tabelle DB\_Customer gelöscht werden und „where“, welche Reihe gelöscht werden soll. Nachdem die Reihe gelöscht ist, kann die neue Reihe mit den korrigierten oder neuen Werten eingefügt werden.   
Bei der Funktion einer Reihe zu löschen wird die gleiche Methode ausgeführt, wie beim schreiben. Jedoch wird der Befehl „conn.Insert(row);“ nich ausgeführt. Somit wird nur die Reihe gelöscht und keine neue eingefügt.   
Die dritte grundlegende Funktion ist das Lesen aus den Tabellen. Hierfür muss wieder eine Verbindung zu der Datenbank SQ Lite aufgebaut werden. Query heißt die „Lesefunktion“ in C#. Query hat seinen Ursprung aus dem Lateinischen (quaerere) und bedeutetet übersetzt fragen/suchen. (centron, 2023) Dieses „query“ muss definiert werden „wo“ gesucht werden soll. „Where“definiert in diesem Fall wo gesucht werden soll. Wenn die ganze Tabelle ausgegeben werden soll wird „where“ kein Inhalt zugewiesen where=““. Anhand dem definierten“query“ kann die Suchfunktion ausgeführt werden. Ist diese fertiggestellt, wird die Verbindung zur Datenbank wieder getrennt.

*//Initialisierung verbindung*SQLiteConnection conn = null;  
  
try  
{  
 *//Aufbau Verbindung mit Datenbank* conn = new SQLiteConnection(dataSource);  
 *//Zusammensetzung des Querys für die Suche  
 //Name der Tabelle in Query aufnehmen* string query = $"SELECT \* FROM {nameof(Db\_Customer)} ";  
 if (where != "")  
 {  
 *//Suchoption* query += "WHERE " + where;  
 }  
 *//Suchen anhand des gegebenen Parameters* return (conn.Query<Db\_Customer>(query), null);  
}  
catch (Exception ex)  
{  
 return (new List<Db\_Customer>(), new Error(ex.ToString()));  
}  
finally  
{  
 conn?.Close();  
}

Wenn dem „query“ einen zu suchenden Wert zugewiesen werden soll, wird dies über eine weiter Methode „Readwith“ getan, um Beispiel der „ReadwithID“ Methode. In dieser Methode wird „where“ den Wert der zu suchenden ID zugewiesen. Nachdem der Wert zugewiesen wurde, springt das Programm mit dem „return“ Befehl zu der Methode „Read“ und führt diese aus. In dieser wird dann „query“ den Wert von „where“ zugewiesen.

public static (List<Db\_Customer>, Error) ReadwithID(string ID, string dataSource)  
{  
 var where = $"ID='{ID}'";   
 return Read(where, dataSource);  
}

Der Anwender kann zum Beispiel einen neuen Kunden anlegen mit dem Button „+Add New Customer“. Daraufhin öffnet sich ein Popup Fenster, in welchem alle relevanten Informationen zu dem Kunden eingetragen werden. Sind nicht alle Felder ausgefüllt, ploppt eine Fehlermeldung auf, welchen den Anwender auffordert die restlichen Daten abzulegen. Das Design ist immer das selbe zum Erstellen von neuen Aufträgen, Kunden, Ressourcen oder Bestellungen. Somit findet sich der Anwender schnell in der Handhabung zurecht.   
Ein Bild, das Screenshot, Text, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Benutzerverwaltung

Die Benutzerverwaltung spielt bei der Timetracking App eine wichtige Rolle. Für jede Aktion in der Anwendung ist eine Anmeldung notwendig. Erfolgt diese nicht, erscheint eine Fehlermeldung und fordert den Benutzer auf sich anzumelden.

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

Durch eine konsequente Anmeldung ist immer gewährleistet, dass nur die Funktionen, welche für den Benutzer bestimmt sind, ausgeführt werden können. Nur ein Administrator kann einen neuen Benutzer anlegen oder löschen. Für das Anlegen wird dem neuen Benutzer eine ID und eine Rolle zugeteilt. Des Weiteren werden Benutzern auch deren „Rechte“ in der App zugewiesen: „Read only Permission“, „Write Permission“ und „Full Acess“.

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

Automatisch generierte Beschreibung

* Entwicklung
  + Technologiewahl
    - App IOS und Android nur mit einem Tool entwickeln
  + Planung (Entity Realtionship Diagramm, Planung Datenbank: Tabellen (Inhalt), Prozesse / Funktionen planen)
  + Software design
    - Wpf mvvm <https://www.youtube.com/watch?v=JWCudlf5c4Q> (Theorie schreiben)
    - Error-handling (Ausgabe Messagebox) 🡪 David
    - Aufbau Datenbank (db\_Klassen rw\_klassen für eine Tabelle zuständig) 🡪 Lauritz
    - Socketserver 🡪 Sean
    - PDF
    - Tabellenansicht WPF
    - Benutzerverwaltung

# Fazit und Ausblick

Beispiel and DB\_Customer / Rw\_Customer

Aufbau DB:

1. Erstellen einer DB in Rider (Abspeichern der .sqlite Datei an gewünschtem Ort, am besten im Debug/Kompilierungsordner(Unabhängigkeit vom Instaltionsort)), alle Schemas aktivieren für Ansicht der später erstellten Tabellen
2. Verweisen auf DB-Datei mit relativem Pfad ausgehend von der .exe datei
3. Ansicht im Rider aktivieren
4. Erstellen von Klassen.
5. Eine Klasse für Tabellenerstellung/Eigenschaften (bei uns DB\_NamedesTabellentyps)
   1. Zuweisung Propertys (Spaltenname und Datentyp (bool,string etc.)) Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

      Automatisch generierte Beschreibung
   2. Erstellung einer Methode zur Erstellung der Tabelle Ein Bild, das Text, Screenshot, Software, Schrift enthält.

      Automatisch generierte Beschreibung



Try catch (Versucht das was im try steht zu tun, wenns nicht geht dann catch)

* 1. Erstellung einer Klasse zum Lesen und Schreiben in die jeweilige Tabelle (Rw\_ NamedesTabellentyps)
  2. Drei Grundsätzliche funktionen (Write,Delete und Read)
  3. Write: Ein Bild, das Text, Screenshot, Software enthält.

     Automatisch generierte Beschreibung
  4. Delete ist das gleiche nur ohne conn.Insert(row);
  5. Read Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

     Automatisch generierte Beschreibungbei aufruf von Read mit leerem string where also: ““ werden alle Reihen in der Datenbank zurückgegeben.
  6. Erstellen der Readwith Methoden Ein Bild, das Text, Screenshot, Schrift enthält.

     Automatisch generierte Beschreibung Hier wird gezielt nach einem Inhalt gesucht z.B. der ID 1. Methoder ruft immter Read auf und übergibt den Suchparameter.

1. Aufruf der Funktionen: (Popups/EditCustomer.xaml.cs)
   1. Write/DeleteEin Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

      Automatisch generierte Beschreibung
   2. ReadwithEin Bild, das Text, Screenshot, Schrift, Reihe enthält.

      Automatisch generierte Beschreibung
   3. Read (Customers.xaml.cs) Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

      Automatisch generierte Beschreibung
2. Eventuell noch beschreibung der DB Ansicht / Editors (manuelles Ändern von Werten in einer Tabelle)

Benutzerverwaltung:

* Jede Seite benötigt eine Anmeldung

Struktur:

Basisseite: Ordner View

Edit\_Field: Ordner Popups

Delete\_Field: Ordner Popups

Bei Finish Order gibts noch nen Finish Popup

# Abbildungsverzeichnis

1. Tabellenverzeichnis

1. Literaturverzeichnis

[Windows Presentation Foundation (WPF) - Begriffserklärung im Entwickler-Lexikon/Glossar auf www.IT-Visions.de](https://www.it-visions.de/glossar/alle/3718/Windows_Presentation_Foundation.aspx)