**Naukanu Sailing School**

**Endbericht**

Dokumenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor(en) | Kommentar / Beschreibung |
| 0.1 | 07.02.2014 | Benjamin Böcherer, Stefan Müller, Dominik Schumacher, Tobias Meyer | Initiale Erstellung |
| 0.2 | 13.03.2014 | Benjamin Böcherer, Stefan Müller, Dominik Schumacher, Tobias Meyer | Korrektur |
| 0.3 | 20.03.2014 | Benjamin Böcherer, Stefan Müller, Dominik Schumacher, Tobias Meyer | Finalisierung |

Inhalt

[1 Verwendete Technologien 4](#_Toc391792635)

[1.1 Microsoft .NET-Framework 4](#_Toc391792636)

[1.2 Die Programmiersprache C# 5](#_Toc391792637)

[1.3 XAML 6](#_Toc391792638)

[1.4 Entity Framework 6](#_Toc391792639)

[1.5 Modern UI 7](#_Toc391792640)

[1.6 MVVM 7](#_Toc391792641)

[1.7 Microsoft SQL Server 8](#_Toc391792642)

[2 Softwareentwicklung 9](#_Toc391792643)

[2.1 Vorgehensmodell Scrum 9](#_Toc391792644)

[2.2 Prototyping 10](#_Toc391792645)

[2.3 Qualitätsmanagement 12](#_Toc391792646)

[2.4 Organisationswerkezeuge 14](#_Toc391792647)

[2.5 Datenbank 16](#_Toc391792648)

[2.6 UML 16](#_Toc391792649)

[2.7 Quellcodeverwaltung 17](#_Toc391792650)

# Verwendete Technologien

Der Kern der Anwendung „Naukanu Sailing School Manager“ basiert auf Microsoft Technologien im .NET-Umfeld. Die Daten werden in dem Datenbankserver Mircrosoft SQL Server persistiert. Die graphische Benutzeroberfläche wurde ebenfalls mit .NET-Technologien umgesetzt. Zusätzlich wird hier noch das UI-Framework „Modern UI“ eingesetzt, um die Oberfläche grafisch an aktuelle Softwareprojekte anzupassen. Die eingesetzten Technologien stehen kostenlos zur Verfügung. Davon profitiert auch die Segelschule als Kunde, die nur die reinen Entwicklungskosten der Anwendung und keine weiteren Lizenzkosten zahlt.

Die relevanten Technologien werden in dem folgenden Kapital genauer erläutert.

## Microsoft .NET-Framework

Die Anwendung „Naukanu Sailing School Manager“ benötigt das Microsoft .NET-Framework ab der Version 4.5. .NET bezeichnet eine von Microsoft entwickelte Software-Plattform zur Entwicklung und Ausführung von Anwendungsprogrammen. Es besteht aus einer Laufzeitumgebung (Common Language Runtime), in der die Programme ausgeführt werden, sowie einer Sammlung von Klassenbibliotheken, Programmierschnittstellen und Dienstprogrammen. .NET ist auf verschiedenen Plattformen verfügbar und unterstützt die Verwendung einer Vielzahl von Programmiersprachen. Die .NET-Programme werden zunächst in eine Zwischensprache (Common Intermediate Language) übersetzt, bevor sie von der Laufzeitumgebung ausgeführt werden. Diese Übersetzung geschieht in der Regel mithilfe eines sogenannten Just-In-Time-Compilers. Mit .NET löste Microsoft zuvor eingesetzte Softwareentwicklungskonzepte wie das Component Object Model (COM) ab.

Die folgende Tabelle gibt eine Übersicht der verschiedenen .NET-Versionen:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Versionsnummer** | **Datum** | **enthalten in** |
| 1.0 | 1.0.3705.0 | 5. Januar 2002 | n/a |
| 1.1 | 1.1.4322.573 | 1. April 2003 | Windows Server 2003 |
| 2.0 | 2.0.50727.42 | 7. November 2005 | Windows Server 2003 R2 |
| 3.0 | 3.0.4506.30 | 6. November 2006 | Windows Vista, Windows Server 2008 |
| 3.5 | 3.5.21022.8 | 9. November 2007 | Windows Server 2008 R2 |
| 3.5 SP 1 | 3.5.30729.1 | 11. August 2008 | Windows 7 mit SP1 |
| 4.0 | 4.0.30319 | 12. April 2010 | n/a |
| 4.5 | 4.5.50501 | 15. August 2012 | Windows 8, Windows Server 2012 |
| 4.5.1 | 4.5.50938 | 12. Oktober 2013 | Windows 8.1, Windows Server 2012 R2 |

Die folgende Grafik gibt einen Überblick die verschiedenen Bausteine, die kontinuierlich in das .NET-Framework eingebaut wurden:



## Die Programmiersprache C#

C# (lies englisch c sharp) ist eine von Microsoft im Rahmen seiner .NET-Strategie entwickelte Programmiersprache und ist bei ECMA und ISO als Standard registriert.

C# greift Konzepte der Programmiersprachen Java, C++, Haskell, C sowie Delphi auf und zählt zu den objektorientierten Programmiersprachen. Es unterstützt sowohl die Entwicklung von sprachunabhängigen .NET-Komponenten als auch COM-Komponenten für den Gebrauch mit Win32-Anwendungsprogrammen.

Wir haben uns für diese Programmiersprache entschieden, da wir eine Windows-Desktop-Anwendung entwickeln wollte. Aufgrund von Vorkenntnissen der Projektteilnehmer im .NET-Umfeld (C#, VB .NET) bat sich diese Technologie hervorragend an.

## XAML

XAML ist die Abkürzung für „Extensible Application Markup Language“ und ist eine allgemeine Beschreibungssprache für die Oberflächengestaltung von Anwendungen die von Microsoft entwickelt wurde. Bei der Beschreibungssprache handelt es sich um eine XML-basierte Sprache. Es lassen sich damit grafische Elemente, Benutzeroberflächen, Verhaltensweisen, Animationen, Transformationen, Darstellung von Farbverläufen, Abspielen von Mediadateien und vieles mehr definieren. Bei einer üblichen Verwendung werden XAML-Dateien von optisch orientierten Design- und Entwicklerwerkzeugen wie Microsoft Expression Blend, Microsoft Visual Studio oder XAML Pad generiert.

Mit XAML kann man unter Verwendung jeder beliebigen anderen Programmierschnittstelle Anwendungen entwickeln, wobei XAML eine eigenständige Sprache darstellt. Der Hauptaspekt dieser Technologie ist die verringerte Komplexität, die Programme haben müssen, um XAML zu verarbeiten, weil es sich im Grunde um einfaches XML handelt. XAML-Dateien sind hierarchisch strukturiert. Ein oder mehrere Elemente können, abhängig von ihrer Ordnung, das Layout und Verhalten der Oberfläche beeinflussen. Jedes Element besitzt nur ein Elternelement. Jedes Element kann eine unbegrenzte Anzahl von Kindelementen besitzen, nur bei einigen wenigen ist die Anzahl eingeschränkt. In allen XAML-Anwendungen ist das Wurzelobjekt typischerweise ein Panel (oder eines seiner sechs Unterarten), das sich um Positionierung und Rendern jeglichen Inhaltes kümmert.

Eigenschaften und Einstellungen z. B. einer Schaltfläche werden wie bei XML bzw. HTML im Tag als Attribute aufgeführt. Wie jede XML-Datei besitzt XAML ein Wurzel-Tag. Bei XAML nimmt man ein Vaterobjekt (z. B. ein <Window>), in das man weitere Kinder (also Elemente) einfügen kann.

## Entity Framework

Das ADO.NET Entity Framework (kurz: ADO.NET EF) ist ein ORM-Framework von Microsoft, welches auf ADO.NET basiert. Objektrelationale Abbildung (englisch object-relational mapping, ORM) ist eine Technik der Softwareentwicklung, mit der ein in einer objektorientierten Programmiersprache geschriebenes Anwendungsprogramm seine Objekte in einer relationalen Datenbank ablegen kann. Dem Programm erscheint die Datenbank dann als objektorientierte Datenbank, was die Programmierung erleichtert. Implementiert wird diese Technik normalerweise mit Klassenbibliotheken, wie beispielsweise ADO.NET Entity Framework, Hibernate für die Programmiersprache Java oder SQLAlchemy für Python.

Im einfachsten Fall werden Klassen auf Tabellen abgebildet, jedes Objekt entspricht einer Tabellenzeile und für jedes Attribut wird eine Tabellenspalte reserviert. Die Identität eines Objekts entspricht dem Primärschlüssel der Tabelle. Hat ein Objekt eine Referenz auf ein anderes Objekt, so kann diese mit einer Fremdschlüssel-Primärschlüssel-Beziehung in der Datenbank dargestellt werden.

Die „Naukanu Sailing School Manager“-Anwendung setzt das Entity Framework in der Version 6.0.0 ein.

## Modern UI

Für die grafische Oberfläche haben wir uns für ein an Windows 8 angelehntes Aussehen entschieden. Um nicht alle optischen Features neu entwickeln zu müssen, wurde das UI-Framework „Modern UI“ eingesetzt. Damit lassen sich z.B. die aus Windows 8 bekannten „Circle-Buttons“ oder „Live-Tiles“ leichter einbinden. Das Framework steht kostenlos zur Verfügung.

## MVVM

Model View ViewModel (MVVM) ist eine Variante des Model View Controller-Musters (MVC) zur Trennung von Darstellung und Logik der Benutzerschnittstelle (UI). Es zielt auf moderne UI-Plattformen wie Windows Presentation Foundation (WPF), Silverlight und HTML5 ab. MVVM sieht eine Rollentrennung von UI-Designern und Entwicklern vor, wodurch Anwendungsschichten von verschiedenen Arbeitsgruppen entwickelt werden können. Designer können einen Fokus auf User Experience setzen und Entwickler die UI- und Geschäftslogik schreiben.

Das MVVM wurde 2005 von Microsoft MVP John Gossman veröffentlicht. Es ist eine Spezialisierung des Presentation Model von Martin Fowler und ist mit diesem insofern identisch, als beide Muster Zustand und Verhalten der View in ein separates UI-Model (Presentation bzw. View Model) verschieben. Das Presentation Model ermöglicht allerdings das Erzeugen einer View unabhängig von der UI-Plattform, wohingegen das MVVM ursprünglich auf UIs mittels WPF abzielt. Es findet allerdings inzwischen auch in anderen Bereichen Anwendung, ähnlich wie bei MVC.

Das MVVM nutzt die funktionale Trennung des MVC und Datenbindung, um eine lose Kopplung zu erreichen. Es beinhaltet drei Komponenten, wobei Model und View denen des klassischen MVC ähneln:

* **Model**: Datenzugriffsschicht für die Inhalte, die dem Benutzer angezeigt und von ihm manipuliert werden. Dazu benachrichtigt es über Datenänderungen und führt eine Validierung der vom Benutzer übergebenen Daten durch. Es beinhaltet die gesamte Geschäftslogik und ist für sich alleine durch Unit Tests überprüfbar.
* **View:** Alle durch die Grafische Benutzeroberfläche (GUI) angezeigten Elemente. Es bindet sich an Eigenschaften des ViewModel, um Inhalte darzustellen und zu manipulieren sowie Benutzereingaben weiterzuleiten. Durch die Datenbindung ist die View einfach austauschbar und ihr Code-Behind gering.
* **ViewModel**: beinhaltet die UI-Logik (Model der View) und dient als Bindeglied zwischen View und obigem Model. Einerseits tauscht es Information mit dem Model aus, ruft also Methoden oder Dienste auf. Andererseits stellt es der View öffentliche Eigenschaften und Befehle zur Verfügung. Diese werden von der View an Steuerelemente gebunden, um Inhalte auszugeben bzw. UI-Ereignisse weiterzuleiten. Insgesamt wird CRUD ermöglicht. Das ViewModel darf dabei keinerlei Kenntnis der View besitzen.

In Bezug auf WPF bedeutet MVVM, dass die View aus rein deklarativem XAML-Markup besteht. Sie kann separat von UI-Designern entworfen werden, wobei lediglich Datenbindungen zum ViewModel erzeugt werden müssen, aber kein Code-Behind. Die Logik wird in einer Programmiersprache wie C# implementiert. Die Abhängigkeiten zwischen Markup und Code werden minimiert.

Die Menge an Geschäftslogik im Code-Behind der View wird reduziert. Dadurch können UI-Designer Views rein gestalten während Entwickler unabhängig davon die Models und ViewModels implementieren. Des Weiteren sind – die Korrektheit der Datenbindung vorausgesetzt – keine (in der Regel manuellen) UI-Tests nötig. Stattdessen genügen codebasierte Modultests des ViewModel. Zuletzt „erbt“ MVVM von MVC die leichtere Austauschbarkeit der View.

## Microsoft SQL Server

Der SQL Server ist ein relationales Datenbankmanagementsystem, das sich am Standard der aktuellen SQL-Version orientiert. Der Microsoft SQL Server liegt in verschiedenen Editionen vor, die ein vielfältiges Angebot abdecken. Die Editionen unterscheiden sich vor allem im Preis, ihren Funktionen und Hardwareeinschränkungen. Der SQL Server besteht aus vielen Services und Tools, wie Analysis Services, Reporting Services, Integration Services und Sync Services.

Microsoft SQL Server verwendet für Datenbankabfragen die SQL-Variante T-SQL (Transact-SQL). T-SQL fügt hauptsächlich zusätzliche Syntax zum Gebrauch in Stored Procedures und Transaktionen hinzu. Weiterhin unterstützt er OLE DB und ODBC (Open Database Connectivity).

In diesem Projekt kommt der SQL Server in der Version 2008 R2 Express zum Einsatz. Diese Variante steht kostenlos zur Verfügung, hat aber folgende Einschränkungen:

* Es wird nur ein Prozessor bzw. ein Prozessorkern verwendet.
* Die Express-Edition nutzt maximal 1 GB Arbeitsspeicher.
* Eine Datenbank darf maximal 10 GB groß sein.

# Softwareentwicklung

Für die Umsetzung der Anwendung musste vorab eine generelle Struktur aufgebaut werden. Nachdem die ersten Entscheidungen im Bereich der zu verwendenden Technologien geklärt wurden, ging es um die konkrete Entwicklung der Software. Wir haben uns dazu entschieden, als Vorgehensmodell Scrum einzusetzen und zunächst einen ersten Prototyp zu entwickeln, um darauf aufbauend die Anwendung abzuschließen. Die folgenden Abschnitte erklären die Prozesse detailliert.

## Vorgehensmodell Scrum

Der Ansatz von Scrum ist empirisch, inkrementell und iterativ. Er beruht auf der Erfahrung, dass die meisten modernen Entwicklungsprojekte zu komplex sind, um durchgängig planvoll umgesetzt zu werden, und auf der Erkenntnis, dass allein ständig verfügbares Feedback den Erfolg sichert. Damit wird vermieden, die anfänglich gegebene Komplexität durch einen komplexeren Plan zu steigern.

Ziel ist die schnelle, kostengünstige und qualitativ hochwertige Fertigstellung eines Produktes, das einer zu Beginn formulierten Vision entsprechen soll. Scrum kennt drei Rollen für direkt am Prozess Beteiligte: Product Owner (stellt fachliche Anforderungen und priorisiert sie), ScrumMaster (managt den Prozess und beseitigt Hindernisse) und Team (entwickelt das Produkt). Daneben gibt es als Beobachter und Ratgeber noch die Stakeholders.

Die Anforderungen (Requirements) werden in einer Liste (Product Backlog) gepflegt, erweitert und priorisiert. Das Product Backlog ist ständig im Fluss. Um ein sinnvolles Arbeiten zu ermöglichen, wird monatlich vom Team in Kooperation mit dem Product Owner ein definiertes Arbeitspaket dem oberen, höher priorisierten Ende des Product Backlogs entnommen und komplett in Funktionalität umgesetzt (inkl. Test und notwendiger Dokumentation). Dieses Arbeitspaket, das Increment, wird während der laufenden Iteration, des sog. Sprints, nicht durch Zusatzanforderungen modifiziert, um seine Fertigstellung nicht zu gefährden. Alle anderen Teile des Product Backlogs können vom Product Owner in Vorbereitung für den nachfolgenden Sprint verändert bzw. neu priorisiert werden.

Das Arbeitspaket wird in kleinere Arbeitspakete (Tasks) heruntergebrochen und mit jeweils zuständigem Bearbeiter und täglich aktualisiertem Restaufwand in einer weiteren Liste, dem Sprint Backlog, festgehalten. Während des Sprints arbeitet das Team konzentriert und ohne Störungen von außen daran, die Tasks aus dem Sprint Backlog in ein Increment of Potentially Shippable Functionality, also einen vollständig fertigen und potentiell produktiv einsetzbaren Anwendungsteil, umzusetzen.



## Prototyping

Das Prototyping ist eine Methode der Softwareentwicklung, die vor dem Beginn der Entwicklungsphase die benötigten Risiken aufzeigt bzw. diese egalisiert und zudem schnell zu ersten Ergebnissen führt und frühzeitiges Feedback bezüglich der Eignung eines Lösungsansatzes ermöglicht. Dadurch ist es möglich, Probleme und Änderungswünsche frühzeitig zu erkennen und mit weniger Aufwand zu beheben, als es nach der kompletten Fertigstellung möglich gewesen wäre.

Die verschiedenen Arten des Prototyping werden im Folgenden näher erläutert

* **Exploratives Prototyping**

Erstes Ergebnis: Eine übersichtliche Anforderungsspezifikation

Ziel: Ziel ist es nachzuweisen, dass Spezifikationen oder Ideen tauglich sind.

Das explorative Prototyping wird zur Bestimmung der Anforderungen und zur Beurteilung bestimmter Problemlösungen verwendet und konzentriert sich dabei auf die Funktionalitäten des Systems.

* **Evolutionäres Prototyping**

Erstes Ergebnis: Ein Programm mit den Grundfunktionalitäten

Ziel: Anhand der Grundfunktionalitäten die Akzeptanz beim Nutzer und die Notwendigkeit ergänzender Funktionen zu überprüfen

Beim evolutionären Prototyping wird die Anwendung nach und nach erweitert. Dabei wird vor Allem das Feedback der zukünftigen Nutzer bzw. des Auftraggebers genutzt. Der Prototyp wird dabei stets lauffähig gehalten und bis zur Produktreife weiterentwickelt.

* **Experimentelles Prototyping**

Erstes Ergebnis: ein erster experimenteller Prototyp

Ziel: Sammeln von Erfahrungen mit dem Prototyp

Bei diesem Vorgehen wird zu Forschungszwecken bzw. zur Suche nach Möglichkeiten zur Realisierung ein experimenteller Prototyp entwickelt. An diesem wird anschließend eine sehr umfangreiche Problemanalyse und Systemspezifikation durchgeführt. Die gewonnenen Erkenntnisse können anschließend in einem richtigen Produkt verwertet werden.

* **Rapid Control Prototyping**

Rapid Control Prototyping bezeichnet die Softwareentwicklung von Regelungen und Steuerungen, mit Hilfe grafischer Tools.

* **Vertikales Prototyping (Durchstich)**

Erstes Ergebnis: Ein ausgewählter Teil des Systems wird durch alle Ebenen hindurch implementiert.

Ziel: Bestrebung explizit einen konkreten Teil eines Programms anzufertigen.

Hierbei wird ein ausgewählter Teil umgesetzt. Dies eignet sich besonders für Fälle, in denen noch Funktionalitäts- oder Implementierungsfragen ungeklärt sind. Abgeschlossene Teile können dann bereits umgesetzt werden, bevor die Anforderungen für den Rest komplett festgelegt wurden.

* **Horizontales Prototyping**

Erstes Ergebnis: Eine ausgewählte Ebene des Gesamtsystems wird fertiggestellt.

Ziel: Eine funktionierende Ebene, die vorgestellt werden kann, oder an der sich andere Ebenen orientieren können.

In diesem Fall wird nur eine spezifische Ebene des Gesamtsystems realisiert, welche jedoch möglichst vollständig abgebildet wird. (z. B. Realisierung der GUI (Oberfläche) (ohne tiefer liegende Funktionalitäten), zur Vorlage für den Auftraggeber.) Diese Methode hat den Vorteil, dass man dem Auftraggeber schon etwas zeigen kann, ohne das komplette System entwickelt zu haben. Dies setzt jedoch eine starke (sowieso sinnvolle) Trennung der einzelnen Komponenten voraus. Die Oberfläche muss dementsprechend unabhängig von der dahinter liegenden Logik funktionieren oder wenn die Logik-Ebene umgesetzt wird, muss sie unabhängig von der Oberfläche funktionieren.

Wir haben uns für einen evolutionären Prototyp entschieden. Somit konnte jeder Projektteilnehmer sein Feedback zu neuen Funktionen oder optischen Anpassungen abgeben.

## Qualitätsmanagement

Die technischen und projektorganisatorischen Projektteilnehemer verwendeten das zuvor vorgestellte SCRUM Vorgehensmodell in Verbindung mit dem „Continuous Integration“ (CI) Ansatz.

Dabei erstellt der Entwickler zunächst in seiner lokalen Arbeitsumgebung den Quellcode, der innerhalb der Entwicklungsumgebung (IDE) kompiliert und lokal installiert wird. Zuvor und sukzessive schreibt er für die fachlichen und nicht-fachlichen Anforderungen einen oder mehrere Unit Tests und führt diese aus (z. B. TestNG, JUnit, MSUnit etc.). Dies kann innerhalb der IDE oder als separater Build (Maven, Ant, MSBuild) geschehen. Das Implementieren von Komponententests dient der. Werden alle Unit Tests erfolgreich in der lokalen Umgebung ausgeführt, so „checkt“ der besagte Entwickler seine Änderungen in den zentralen Quellcodeserver ein (GitHub).

Zu vorgegebener Zeit greift nun ein sog. CI-Server (Team Foundation Server) dieses Repository ab und führt ein „Check-Out“ durch, so dass der gesamte Quellcode aller Entwickler nun innerhalb des Servers vorliegt. Danach werden die entsprechenden Quellcodedateien kompiliert und zusammen mit eventuell vorhandenen Ressourcendateien zu Artefakten gebunden (exe, dll etc.). Zudem werden nun die bestehenden Tests mittels der CI-Engine ausgeführt.

Durch geeignete Schwellenwerte kann hier der CI-Server entscheiden, ob es sinnvoll und lohnenswert erscheint, ein neues Release zu erstellen. Typische Kennzahlen hierfür sind:

* Sind alle fachlichen Tests positiv verlaufen?
* Haben alle Negativ-Tests das gewünschte Ergebnis erzielt?
* Konnten die Performance-Tests innerhalb des designierten Zeitrahmens ausgeführt werden?
* Wurden mindestens 75 % der funktionalen Anforderungen durch Tests abgedeckt?
* Wurden alle erforderlichen Formatierungsregeln eingehalten?
* Gibt es keine offensichtlichen Fehler im Quellcode?

Können alle oben angegebenen Fragen positiv beantwortet werden, so wird ein Release erstellt und automatisch in der Testumgebung installiert. Somit steht nun ein neues Release (Zwischenergebnis) für „Beta-Tester“ bzw. Kunden zur Verfügung. Entwickler und Kunden haben dadurch eine sehr konkrete Vorstellung vom aktuellen Entwicklungsstand und den letzten umgesetzten Änderungen (direkte Aktions-Reaktions-Analyse). Basierend auf diesem Release kann nun ein gezielter Dialog geführt werden, wenn es darum geht, Anforderungen anzupassen oder zu erweitern.

Zusammengefasst ergeben sich aus dem Konzept „Continuous Integration“ folgende Vorteile:

* Umsetzung der Release-Often-Paradigmen der agilen Softwareentwicklung
* Zentrale Quellcodeversionierung, Möglichkeit des „Zurückspringens“ auf ältere Versionen
* Zeitnahes Testen unter „Realbedingungen“. Wenn signifikante fachliche Fehler oder Performance Probleme auftreten, kann die Ursache schnell gefunden werden, da zwischen zwei Releases wenige Änderungen stattfinden.
* Fach- und Performancetests als integraler Bestandteil der ganzheitlichen Softwareentwicklung. Keine funktionale Anforderung wird eingecheckt ohne zugehörigen Test.
* Identifikation der Testabdeckung. Wenn beispielsweise 95% des Gesamt-Quellcodes getestet ist und keine Fehler auftraten, dann kann sehr sicher davon ausgegangen werden, dass die Software macht, was sie soll.
* Identifikation von Hot Spots und Bottlenecks bei jedem Release. Welche Methoden werden besonders oft aufgerufen, welche nehmen absolut und relativ am meisten Zeit ein? Wo ist somit Optimierungspotential?
* Prüfung, ob vorgegebene Quellcodemetriken und Dokumentationsregeln eingehalten wurden.
* Identifikation von offensichtlichen Programmierfehlern.
* Automatische Benachrichtigungsfunktion beim Erstellen des Release-Artefakts (E-Mail an Fachbereich, falls neues Release vorhanden, E-Mail an Entwicklerkreis im Falle eines Fehlers).



## Organisationswerkezeuge

Zur Erfassung der fachlichen Vorgaben (User Stories), Aufgaben, Releases und identifizierten Bugs wird ein so genanntes Application Lifycycle Management (ALM)-System eingesetzt (Microsoft Team Foundation Server). ALM ist eine Kombination aus der Entwicklung und Betreuung von Applikationen (Anwendungssoftware) über deren gesamten Lebenszyklus. Dies beinhaltet auch eine umfassende Anwenderbetreuung (Support) und die Weiterentwicklung der Software. Somit können zu jeder Zeit Informationen über den aktuellen Projektstand ermittelt werden, die in aller Regel über folgende Kennzahlen definiert werden:

* Welche Aufgaben / User Stories / Bugs sind noch offen, in Bearbeitung, fertig und geprüft in dem aktuellen Release?
* Welche Aufwände wurden bereits geleistet und welche Aufwände stehen für das aktuelle Release noch aus?
* Können die noch ausstehenden Aufwände in der noch zur Verfügung stehenden Zeit geleistet werden?
* Gibt es eine Diskrepanz zwischen der ursprünglich geschätzten und tatsächlich benötigten Zeit?



Durch die Visualisierung und Identifikation dieser Kennzahlen ist es kurzfristig möglich, einen objektiven Entwicklungsstand des Softwarevorhabens zu bekommen. Dadurch wird der Projektorganisation die Möglichkeit gegeben, aktiv Ressourcen im Projekt zielorientiert zu steuern. Ist z. B. anhand der noch ausstehenden Tätigkeiten klar, dass in der zur Verfügung stehenden Zeit die Aufgaben nicht abgeschlossen werden können (ausgehend von einem 8 Std. Werktag), kann die Projektleitung nun aktiv Gegenmaßnahmen einleiten (Features aus dem Release herausnehmen, weitere Ressourcen kurzfristig binden, Fertigstellungstermin korrigieren etc.). Somit können zu jeder Zeit authentische Aussagen zur aktuellen Projektlage und den erwarteten Ergebnissen getätigt werden.

Die Dokumentation während der Entwicklung erfolgt in einem Wiki-System (GitHub Wiki), wodurch die Entwickler kollaborativ an der Dokumentation arbeiten und diese kontinuierlich erweitern können.

## Datenbank

Durch den Einsatz des Entity Frameworks gibt es zwei Möglichkeiten, eine Datenbank zu erstellen:

* **Keine Datenbank vorhanden (Code First)**

Bestehende Klassen werden mit sogenannten Annotationen (Table, Column) ausgezeichnet, welche die Abbildung auf eine Datenbank steuern. Darauf aufbauend werden die Datenbank und die Datenbank-Tabellen modelliert und erstellt. Zudem besteht die Möglichkeit ein SQL-Skript zu erstellen, mit dem die Datenbank erstellt wird.

* **Verwendung einer bestehenden Datenbank (Model First)**

Die Entity-Klassen können entsprechend der vorgegebenen Datenbank manuell erstellt, modelliert und ausgezeichnet werden. Mit Hilfe eines Assistenten wird die Datenbank abgefragt und entsprechend der Datenbankstruktur ein passendes Modell erstellt.

Wir haben uns für den Code First-Ansatz entschieden. Es wurden zunächst die notwendigen Klassen (Entitäten) erstellt mit den entsprechenden Beziehungen. Danach wurde das Datenbankmodell mit Hilfe es Entity Framework automatisch erstellt. Bei Bedarf wurde die Datenbank aktualisiert.

## UML

Bei der Erstellung der Klassen wurde UML zur Unterstützung eingesetzt. Die Unified Modeling Language (UML) ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software-Teilen und anderen Systemen. Sie wird von der Object Management Group (OMG) entwickelt und ist sowohl von ihr als auch von der ISO (ISO/IEC 19505 für Version 2.1.2) standardisiert. Folgende Grafik zeigt das UML-Diagramm der fertigen Anwendung:

Bild

## Quellcodeverwaltung

Da der Quellcode aufgrund der räumlichen Trennung der Projektteilnehmer an verschiedenen Orten entwickelt wurde, haben wir uns für die zentrale Quellcodeverwaltung GitHub entschieden. GitHub ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte. Namensgebend ist das Versionsverwaltungs-System Git. Eine Versionsverwaltung ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird. Alle Versionen werden in einem Archiv mit Zeitstempel und Benutzerkennung gesichert und können später wiederhergestellt werden. Somit kann jederzeit nachvollzogen werden, welcher Benutzer welchen Quellcode erstellt oder geändert hat. Ein weiterer Vorteil besteht darin, dass bei Fehlern in der Version zurückgesprungen werden kann.