**Naukanu Sailing School**

**Kursverwaltung**

Dokumenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autor(en) | Kommentar / Beschreibung |
| 0.1 | 07.02.1014 | Benjamin Böcherer, Stefan Müller, Dominik Schuhmacher, Tobias Meyer | Initiale Erstellung |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Inhalt

[1 Prolog 6](#_Toc381645730)

[2 Ausgangssituation 6](#_Toc381645731)

[3 Zielsetzung 6](#_Toc381645732)

[4 Mitwirkung des Auftraggebers 6](#_Toc381645733)

[5 Architektur 7](#_Toc381645734)

[6 Hardware- und Systemvoraussetzungen 8](#_Toc381645735)

[7 Entwicklung der Software 8](#_Toc381645736)

[8 Verwendete Technologien 9](#_Toc381645737)

[9 Features - „Must have“ 9](#_Toc381645738)

[9.1 Benutzerhandbuch 9](#_Toc381645739)

[9.2 Kursverwaltung 9](#_Toc381645740)

[9.3 Mitarbeiterverwaltung 10](#_Toc381645741)

[9.4 Materialverwaltung 10](#_Toc381645742)

[9.5 Kundenverwaltung 10](#_Toc381645743)

[9.6 Erstellung von Rechnungen 11](#_Toc381645744)

[9.7 Entitäten 12](#_Toc381645745)

[9.7.2 Allgemein 12](#_Toc381645746)

[10 Features – „Nice to have“ 12](#_Toc381645747)

[10.1 Bedienbarkeit 12](#_Toc381645748)

[10.2 Integration von E-Mail 12](#_Toc381645749)

[10.3 Benutzer- und Rollenverwaltung 12](#_Toc381645750)

[11 Vorgehensmodell und Qualitätssicherung 13](#_Toc381645751)

[12 Organisationswerkezeuge 17](#_Toc381645752)

[13 Firmenprofil 19](#_Toc381645753)

[14 Mitarbeiterprofile 20](#_Toc381645754)

[15 Aufwandsschätzung 22](#_Toc381645755)

[Signaturen 23](#_Toc381645756)

Die nachfolgend formulierten Bedingungen gelten als Grundlage für die Entwicklung einer Software für die

***„Kursverwaltung“,***

die in Kooperation zwischen

**Naukanu Sailing School**

Musterstraße 15

I-Gardasee

(*Auftraggeber*)

und

**Studs@Work AG**

Musterstraße 1

D-12345 Musterhausen

(*Auftragnehmer*)

entwickelt werden soll.

# Prolog

Die Firma Naukanu Sailing School am Gardasee ist eine Segel- und Surfschule in Norditalien. Die angebotenen Kurse können von Gruppen und Einzelpersonen gebucht werden. Die eingesetzten Kursleiter sind freie Mitarbeiter, die saisonweise beschäftigt und kursweise bezahlt werden. Ein Kursleiter kann dabei für mehr als einen Kurs eingesetzt werden.

# Ausgangssituation

Die Naukanu Sailing School benötigt eine Anwendung für die Kursverwaltung. Diese Anwendung soll folgende Aufgaben bewältigen:

* Verwaltung der freien Mitarbeiter (Vertragsmanagement der freien Mitarbeiter, Bezahlung der freien Mitarbeiter, Aufnahme und Verwaltung der persönlichen und beschäftigungsrelevanten Daten, Einteilung für Kurse)
* Verwaltung der Kurse (Termine, Zuordnung zu Kunden, Gebühren, Bereitstellung des Materials, Kursleiter)
* Materialverwaltung (Einsatzbereitschaft, Aussonderung, Reparaturverwaltung, Neubeschaffung, Daten zu Material wie Merkmale, Marke, Kaufpreis, Reparaturkosten)
* Kundenverwaltung (Daten, gebuchte Kurse, Rechnungserstellung, Zahlungsverfolgung, Mahnwesen)

# Zielsetzung

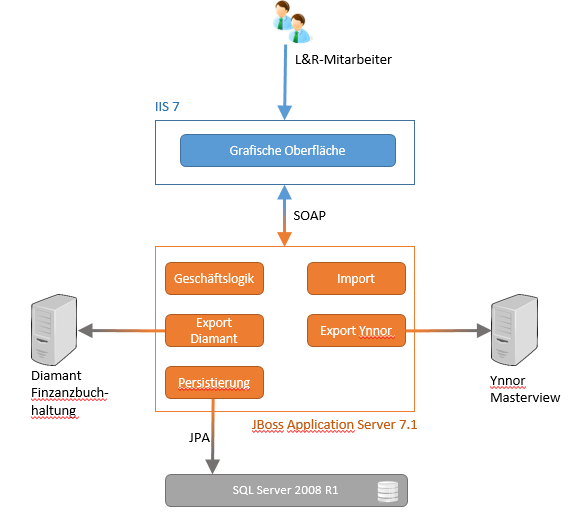
Es soll eine Kursverwaltung erstellt werden. Die Funktionalitäten umfassen die Verwaltung der Stammdaten (Kurse, Teilnehmer, Kursleiter, Material), sowie die Buchhaltung.

# Mitwirkung des Auftraggebers

Der Auftraggeber stellt ab der Auftragserteilung und während der Vertragslaufzeit den Zugriff auf die notwendige Hardware, Software, Schnittstellen, Datenbanken, Räumlichkeiten und Testsysteme mit Testdaten zur Verfügung. Der Auftraggeber ermöglicht den Zugriff auf die Kundendaten und die Aufträge oder stellt vergleichbare Testdaten zur Verfügung. Der Auftraggeber stellt die Daten und Datenstrukturen in konsistenter Form zur Verfügung. Zudem stellt der Auftraggeber virtuelle Maschinen für eine Testumgebung und das Produktivsystem bereit. Der Auftraggeber benennt einen fachkundigen Mitarbeiter als Ansprechpartner, der die entsprechenden Geschäftsinformationen in angemessener Zeit beschaffen kann.

# Architektur

Die folgende Abbildung zeigt die Grobarchitektur des zu entwickelnden Systems:



**Grafische Oberfläche**Die Präsentationsschicht wird durch eine WPF-Anwendung umgesetzt, die mit C# realisiert ist. In diesem Artefakt werden lediglich die grafischen Elemente (Seiten/Formulare) und die für diese Seiten notwendigen Komponenten abgebildet.

**Geschäftslogik**  
Das Geschäftslogikmodul kapselt alle UI-unabhängigen Aktionen, die zur Benutzung der Kernanwendung realisiert werden müssen. Damit ist gewährleistet, dass auch im Falle einer nachträglichen Änderung der Darstellungsschicht die Kernimplementierung nicht maßgeblich geändert werden muss.

**Persistierung**  
Das Persistenzmodul abstrahiert den lesenden und schreibenden Zugriff auf die zentrale Datenbank. Diese Abstraktion vereinfacht beispielsweise einen späteren Austausch des Persistenz-Mechanismus (Entity Framework).

**Datenbank**Die Stud@Works AG hat jahrelange Erfahrung mit verschiedenen Datenbankmanagementsystemen. Daher kann auf eine Open-Source Lösung wie MySQL oder kommerzielle Ansätze wie Microsoft SQL-Server aufgebaut werden. In diesem Projekt kommt ein Microsoft SQL-Server 2008 R2 zum Einsatz.

# Hardware- und Systemvoraussetzungen

Das System soll auf bereits bestehender Hardware betrieben werden. Für das Projekt kommt .Net 4.0 / 4.5 zum Einsatz. Dies setzt die Installation eines entsprechenden Frameworks voraus, das standardmäßig auf jedem aktuellen Windows-Betriebssystem zur Verfügung steht. Der Einsatz von anderen Betriebssystem (Unix, Mac OS) ist nicht vorgesehen.

# Entwicklung der Software

Die Entwicklung der Applikation geschieht in den Räumlichkeiten der Stud@Work AG. Der Auftraggeber erhält lesenden Zugriff auf das GitHub Code-Repository des Projektes.

# Verwendete Technologien

* .NET 4.0/4.5
* C#
* XAML
* LINQ
* Modern UI
* Entity Framework 6
* MVVM

# Features - „Must have“

Im Folgenden werden alle Features beschrieben, die von der Anwendung erfüllt werden müssen.

## Benutzerhandbuch

Ein Benutzerhandbuch soll den Anwender bei der Bedienung der Anwendung unterstützen.

## Kursverwaltung

Der wichtigste Teil der Anwendung, die Kursverwaltung, .soll aus zwei Teilen bestehen:

* Verwaltung des Kurstyps:
  + Hier wird Name des Kurses, Kurzbeschreibung des Inhaltes, notwendiges Material, Anzahl und Qualifikationen der notwendigen Kursleiter, Dauer des Kurses in Stunden und Dauer in Tagen eingegeben (falls z.B. ein Kurs 4 Tage dauert, wobei pro Tag z.B. je 3 Stunden gehalten werden), maximale Teilnehmerzahl pro Kurs, Preis in Euro.
* Leistungsfähige Terminverwaltung, in der die Termine eingegeben geändert und gelöscht werden können. Eine grafische Darstellung wäre hilfreich. Bei Eingabe oder Änderung eines Kurses sollte der (oder die) Kunde(n), Anzahl der Teilnehmer, Kursleiter, und explizit das einzusetzende Material vermerkt werden können.
  + Kurs nach (1) einem Termin nach (2) zu, wobei Terminkollisionen hinsichtlich des Personal- und des Materialeinsatzes zu berücksichtigen sind.
  + Ein Kurs kann nur dann durchgeführt werden, wenn
    - (1) Kursleiter (mit nötigen Lizenzen) vorhanden
    - (2) Material in ausreichender Anzahl vorhanden - und zwar inklusive aller Unterelemente, die für den sicheren Betrieb des Materials nötig sind.

## Mitarbeiterverwaltung

Die Mitarbeiterverwaltung umfasst die Aufnahme der üblichen persönlichen Daten vor, zusätzlich noch die relevanten Qualifikationen (Segel- und Surfscheine, evtl. Lehrberechtigung mit ausstellender Stelle, alternativ Referenzen), Daten eventueller früherer Einsätze bei Naukanu Sailing School und ein „Zufriedenheitsrating“ nach Schulnoten, sowie das Honorar auf Stundenbasis. Zusätzlich sollten noch die Verfügbarkeitszeiten aufgenommen werden können.

Eine Honorarverwaltung (zu zahlende Honorare, gezahlte Honorare, etc.) soll ebenfalls implementiert werden.

## Materialverwaltung

Kursleiter sollen nach Abschluss jedes Kurses einen Materialstatus melden können (z.B. nach Schulnoten von 1 bis 5; 1 uneingeschränkt einsatzbereit, 2=einsatzbereit, 3=eingeschränkt einsatzbereit; 4=nicht einsatzbereit, muss repariert werden, 5=Schrott), und explizit Schäden melden können. Zudem soll die Materialwarte eventuelle Reparaturmaßnahmen eingeben können. De Schulungsleiter können das Material (nach Reparatur) dann wieder höherstufen oder zur Aussonderung empfehlen. Die Materialwarte kann nach erfolgter Reparatur diese im System melden. Hierbei ist ein Ticketsystem hilfreich. Es ist vorzusehen, dass die Materialien in Funktionsgruppen zusammengefasst werden können.

## Kundenverwaltung

Die Kundenverwaltung sollte zwei Arten von Kunden vorsehen:

* Gruppen (z.B. über Reiseveranstalter gebucht)
* Einzelkunden (entweder über Vorbuchung oder als Laufkundschaft).

Es müssen die üblichen Daten des Kunden, sowie gebuchten Kurs (es können auch mehrere sein) und Preis erfasst werden. Eine automatische Erstellung der Buchungsbestätigung/Kursrechnung als Word-, PDF-Dokument oder als E-Mail (optional: alle drei Varianten) soll implementiert werden. Zudem ist ein Mahnwesen mit Zahlungsverfolgung und eine Kundenzufriedenheitsabfrage mit Auswertung nach Kurs und Kursleiter umzusetzen.

## Erstellung von Rechnungen

Der Nutzer muss aus einer Kursteilnahme Rechnungen erzeugen können. Der Nutzer hat dabei die Möglichkeit, einzelne Rechnungspositionen zu erstellen, zu bearbeiten und zu löschen.

Das System berechnet automatisch alle Rechnungsbeträge und Gebühren.

Rechnungen müssen jederzeit vom Nutzer storniert werden können. Das System generiert zu dieser Rechnung dann automatisch eine passende Storno-Rechnung.

Das System vergibt automatisch eine passende Rechnungsnummer für alle Rechnungen. Diese besteht aus den letzten zwei Ziffern des aktuellen Jahres (13,14, usw.) gefolgt von einer fünfstelligen fortlaufenden Nummer. Die gesetzlich geforderte Einmaligkeit der Rechnungsnummern wird vom System sichergestellt.

Das System stellt sicher, dass erstellte Rechnungen, nachdem sie gedruckt wurden, nicht mehr verändert werden können und von Änderungen referenzierter Entitäten nicht länger betroffen sind.

Standardmäßig werden Rechnungen tabellarisch anhand des Leistungsdatums sortiert dargestellt.

Das System muss gewährleisten, eine Rechnung drucken oder in einer PDF-Datei speichern zu können. Dabei muss der Kunde die Möglichkeit haben, einen beliebigen Freitext auf der Rechnung hinzuzufügen. Außerdem müssen die Rechnungen der Briefnorm DIN 5008 entsprechen.

## Entitäten

### Allgemein

Das System muss das Erstellen, Bearbeiten und Löschen folgender Entitäten ermöglichen:   
Kurse, Kunden, Mitarbeiter, Material, Verträge, Termine, Rechnungen, Gutschriften, Provisionen, Stornierungen, Unternehmen, Kontakte.

# Features – „Nice to have“

## Bedienbarkeit

Der Nutzer soll die Anwendung durch verschiedene Tastenkombinationen und den Einsatz der F-Tasten steuern können.

## Integration von E-Mail

Die Anwendung soll eine Anbindung an Microsoft Outlook erlauben, um z. B. per E-Mail Rechnungen oder hinterlegte PDF-Dokumente an Kunden senden zu können.

## Benutzer- und Rollenverwaltung

Der Nutzer kann sich mit einem Benutzer und Passwort an das System anmelden. Danach wird ihm entsprechender seiner Benutzerberechtigung eine angepasste Menüstruktur angezeigt.

# Vorgehensmodell und Qualitätssicherung

Grundlegend kommt als Vorgehensmodel SCRUM zum Einsatz, was aufgrund des iterativ-inkrementellen Ansatzes eine hohe Flexibilität bietet und wenig organisatorischen Zusatzaufwand mit sich bringt. Die Iterationen werden als Sprints bezeichnet. Jeder Sprint resultiert in einem designierten Ergebnis (meist eine neue Version der Software bzw. ein entsprechender Zwischenstand/Meilenstein), was die kontinuierliche und transparente Dokumentation des Projektfortschritts ermöglicht.

Das Vorgehensmodell wird mit einem klassischen SCRUM-Aufbau umgesetzt. Die Rolle des Product Owner wird durch einen Mitarbeiter des Auftraggebers übernommen, die des Scrum Masters durch einen Mitarbeiter der Stud@Works AG, ebenso das selbst organisierende Scrum Team.



Interaktion der Scrum-Mitglieder

Die einzelnen Sprints haben jeweils eine Dauer von drei bis vier Wochen. Zu Beginn eines Sprints findet gemeinsam mit dem Auftraggeber ein Sprint Planning Meeting statt, das auf Wunsch beim Auftraggeber vor Ort durchgeführt wird und in dem die Aufgaben und Ziele für den nächsten Sprint festgelegt werden. Auf diese Weise kann die Software iterativ weiterentwickelt und der Auftraggeber aktiv in den Prozess einbezogen werden. Nach Abschluss eines Sprints führt das Entwickler-Team eine Retrospektive durch, in der die Erfahrungen und Ergebnisse des Sprints reflektiert werden.



Scrumtätigkeiten innerhalb eines Sprints

Während eines Sprints führen die Entwickler der Stud@Works AG tägliche, kurze (max. 15-minütige) Status-Meetings (sog. Daily Scrums) durch, bei dem die Ergebnisse seit dem letzten Treffen und die Planung bis zum nächsten Treffen besprochen werden.

Trotz dieses agilen Vorgehensmodells legt die Stud@Works AG Wert auf eine ausführliche, stets aktuelle und gepflegte Dokumentation von Prozessen und Quellcode.

Die technischen und projektorganisatorischen Mitarbeiter der Stud@Works AG verwenden das zuvor vorgestellte SCRUM Vorgehensmodell in Verbindung mit dem „Continuous Integration“ (CI) Ansatz, sofern vom Auftraggeber kein anderes Vorgehensmodell vorgegeben wird.

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 Qualitätssicherung und ist Bestandteil jeder Softwareentwicklung der Stud@Works AG.

Werden alle Unit Tests erfolgreich in der lokalen Umgebung ausgeführt, so „checkt“ der besagte Entwickler seine Änderungen in einem zentralen Quellcodeserver ein (Subversion, Team Foundation Server, GitHub).

Zu vorgegebener Zeit greift nun ein sog. CI-Server (z. B. Jenkins, 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 (jar, war, ear, rar, exe, dll etc.). Zudem werden nun die bestehenden Tests mittels der CI-Engine ausgeführt. Dies ermöglicht den Einsatz von sogenannten Code Coverage-Werkzeugen. Diese Werkzeuge ermitteln, wie hoch die Testabdeckung des Projekts ist. Weiterhin ist es möglich, durch den Einsatz von statischer Quellcodeanalyse, Inkonsistenzen bei der Benennung und/oder Formatierung des Codes zu identifizieren (i. d. R. Checkstyle, PMD, FindBugs).

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).



CI-Prozess

# Organisationswerkezeuge

Zur Erfassung der fachlichen Vorgaben (User Stories), Aufgaben, Releases und identifizierten Bugs wird ein so genanntes Bug-Tracking-System eingesetzt (Atlassian JIRA in Verbindung mit Atlassian Greenhopper für agile Vorgehensmodelle). 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?



Abbildung: Burndown-Chart zur Ermittlung des Projektstandes

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 (Atlassian Confluence), wodurch die Entwickler kollaborativ an der Dokumentation arbeiten und diese kontinuierlich erweitern können.

Für alle genannten Systeme (JIRA, Confluence, Subversion, Jenkins) ist es auf Wunsch möglich, für den Auftraggeber einen lesenden Zugriff einzurichten. Für die Erfassung und Planung von Prozessen kommen die Standard-Modelle der UML zum Einsatz. Als Werkzeug wird hierbei in der Regel Sparx Enterprise Architect verwendet.



*Screenshot des UML-Modellierungswerkzeuges*

# Firmenprofil

|  |  |
| --- | --- |
| **Firmenname** | Studs@Work AG |
| **Gesellschaftsform** | Aktiengesellschaft |
| **Gegründet** | 1999 |
| **Gründer und Geschäftsführer** | Max Mustermann |
| **Mitarbeiter** | Derzeit 49 |
| **Adresse** | Musterstraße 1  12345 Musterhausen |

Die Studs@Work AG, mit Sitz in Musterhausen, wurde 1999 gegründet und beschäftigt 49 Vollzeitbeschäftigte (davon 46 Entwickler) und 3 Teilzeitbeschäftigte. Die Studs@Work AG ist im Handelsregister von Musterhausen unter der Nummer HRB 12345 eingetragen. Der Fokus der Studs@Work AG liegt auf der Individualsoftwareentwicklung im Enterprise-Bereich, dem Applikationsbetrieb (Betrieb und Support von Enterprise-Applikationen) und der IT-Beratung. Die Enterprise-Applikationen werden in Java Enterprise und Microsoft .NET entwickelt. Zum Kundenkreis gehören u.a. Automobilhersteller, Energielieferanten, Landmaschinenhersteller, Banken, Heiztechnikhersteller sowie diverse IT-Dienstleister und –Systemhäuser. Der Einsatz moderner Arbeitsgeräte ist für die Studs@Work AG selbstverständlich. So setzen unsere Entwickler modernste Notebooks mit Microsoft Windows 7 - 64bit ein, die mit einem Intel Core i5 und mindestens 8GB Arbeitsspeicher ausgestattet sind. Die Notebooks werden spätestens alle 2 Jahre erneuert. Zudem nutzt die Studs@Work AG VMWare-Server (vSphere ESXi) für die Bereitstellung der Entwicklungsumgebungen. Im Java-Umfeld wird Eclipse und im .NET-Umfeld Visual Studio 2010 als Entwicklungsumgebung eingesetzt. Als CI-Server wird Jenkins verwendet, als (Sub)Version Control wird TortoiseSVN genutzt und als Buildtools werden Maven sowie Ant eingesetzt. Die Datenbanksysteme von Oracle, Microsoft SQL, mySQL und PostgreSQL sind bei verschiedenen Kundenprojekten im Einsatz. Für den Einsatz von Qualitätssicherungs-Maßnahmen wird EMMA eingesetzt. Die Projektorganisation wird agil in der Regel mit dem Vorgehensmodell SCRUM oder Adaptionen daraus umgesetzt.

# Mitarbeiterprofile

|  |  |  |  |
| --- | --- | --- | --- |
| **Mitarbeiter** | | **Benjamin Böcherer** |  |
| **Geburtsjahr** | | 1982 | |
| **Ausbildung** | | Fachinformatiker Studium der Wirtschaftsinformatik | |
| **Sprachen** | | Deutsch (Muttersprache)  Englisch | |
| **Tätigkeit** | | Consultant/Developer/Dozent | |
| **Schwerpunkte** | | Realisierung von Java Enterprise Edition Anwendungen mit JSF und EJB  Realisierung von Windows Anwendungen mit dem .NET Framework 3.5 / 4.0  Dozent für Java und .NET-Technologien  Betrieb von Alt-Anwendungen (PHP) | |
| **Projekterfahrung** | **08 / 2013 - heute**  Konzeptionierung und Entwicklung einer Software zur automatischen Bestellabwicklung, die die Verarbeitung von Daten aus unterschiedlichen Quellen (Datenbank, Textdatei) ermöglicht. Diese Bestelldaten werden in einem frei wählbaren Zeitintervall an den Mainframe gesendet. Das Tool läuft als Server-Komponente. Ein entsprechender Client zeigt den Status der Bestellungen an.  Architekt und Entwickler  C#, WPF, SQL Server 2008, Team Foundation Server | | |
|  | **06 / 2012 - aktuell**  Implementierung und Weiterentwicklung eines Portal Systems zur ganzheitlichen Verwaltung von Wertschöpfungs- und Verwaltungsprozessen, inkl. Buchhaltung, Forderungsmanagement, Mitarbeiterverwaltung, Auftragsverwaltung, Angebots-vergabe, Fuhrparkmanagement, Reise- und Hotelbuchung sowie verteilter Zeiterfassung.  Umsetzung im Rahmen eines mehrschichtigen, heterogenen Systems mit Java EE Backend Technologien sowie AJAX-basiertem Web 2.0 MS .NET Frontend sowie der ganzheitlichen Integration von 3rd Party Komponenten mittels Web Services.  *Entwickler*  ASP.NET 4.0, VB.NET, Telerik AJAX Controls, WCF, EJB 3.1, JAX-B, JAX-WS, JBoss 6, , WS-Security, WS-SecureConversation, IIS 7, SQLServer 2008, Scrum | | |
|  | **02 / 2012 - 08 / 2012**  Konzeptionierung und Realisierung einer dynamischen Webanwendung optimiert für die Verwendung von Tablet-PC, um Produktionsgüter auf Großflächen zu lokalisieren und optimal zu lagern. Bei der Einlagerung werden anhand von mehrdimensionalen, dynamisch ermittelten GPS-basierenden Daten anwendungsspezifische Lokalisationsinformationen persistent abgelegt. Bei einer nachfolgenden Disposition und Auslieferung der Güter werden die zuvor erfassten Positionsdaten visuell und kartographisch zur optimierten Auffindung aufbereitet. Der Anwendungskern unterstützt zusätzlich zur optimierten Kommunikation mit den Tablet-Touch-Endgeräten mobile Endgeräte auf Basis von Android und iOS.  *Entwickler*  JBoss Application Server 7.x, Enterprise Java Beans (EJB) 3.1, Java Persistence API (JPA) 2, Context and Dependancy Injection (CDI), Webservices (JAX-WS), Java Server Faces (JSF), PrimeFaces 3.3, Maven 3.04, Microsoft SQL | | |

# Aufwandsschätzung

|  |  |
| --- | --- |
| **Aufgabenbeschreibung** | **Aufwand in Manntagen auf Basis von vier Mitarbeitern** |
| Entwicklungsumgebung einrichten, Testsystem aufbauen, Datenbank-Setup etc. pp. | 3 |
| Architekturaufbau, Projektstruktur aufbauen, CI-System aufbauen | 2 |
| Erstellung der Entitäten und Persistenzkomponenten | 5 |
| Erstellung des Frontend | 15 |
| Erstellung der Rechnungsverwaltung | 15 |
| Erstellung der Kundenverwaltung | 10 |
| Erstellung der Mitarbeiterverwaltung | 10 |
| Erstellung der Kursverwaltung | 10 |
| Erstellung der Terminverwaltung | 20 |
| Erstellung der Materialverwaltung | 10 |
| Summe | 100 |

# Signaturen

|  |  |  |
| --- | --- | --- |
|  | **Datum** | **Unterschrift** |
| **Naukanu Sailing School**  Herr Max Mustermann Geschäftsführer | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
| **Studs@Work AG**  Herr Stefan Müller Entwicklung | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Herr Benjamin Böcherer Entwicklung | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Herr Dominik Schuhmacher Entwicklung | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Herr Tobias Meyer Entwicklung | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |