

IHK-Abschlussprüfung Sommer 2018

**Dokumentation der betrieblichen Projektarbeit**

Für die Abschlussprüfung zum Fachinformatiker - Anwendungsentwicklung

Network Monitoring Tool

Eine Desktop-Applikation

zur Überwachung der Verfügbarkeit von Servern

**Prüfungsbewerber**

Christoph Kiank

Lämmersieth 54

22305 Hamburg

**Prüflingsnummer**

**131 54036**

|  |  |
| --- | --- |
| **Praktikumsbetrieb:** | BITMARCK Technik GmbH  Hammerbrookstraße 38  20097 Hamburg |

|  |  |
| --- | --- |
| **Ausbildungsstätte:** | CBW College Berufliche  Weiterbildung GmbH  Frankenstraße 3  20097 Hamburg |

Abgabetermin: Hamburg, den 04.05.2018

BITMARCK_Logo_4c_Bb

Inhaltsverzeichnis

[Abbildungsverzeichnis III](#_Toc513155506)

[Tabellenverzeichnis IV](#_Toc513155507)

[Abkürzungsverzeichnis V](#_Toc513155508)

[1 Einleitung 1](#_Toc513155509)

[1.1 Projektumfeld 1](#_Toc513155510)

[1.2 Projektbegründung 1](#_Toc513155511)

[1.3 Projektziel 2](#_Toc513155512)

[1.4 Zielgruppe 2](#_Toc513155513)

[2 Projektplanung 3](#_Toc513155514)

[2.1 Projektphasen 3](#_Toc513155515)

[2.2 Ressourcenplanung 3](#_Toc513155516)

[2.3 Entwicklungsprozess 4](#_Toc513155517)

[3 Analysephase 5](#_Toc513155518)

[3.1 Ist-Analyse 5](#_Toc513155519)

[3.2 Wirtschaftlichkeitsanalyse 5](#_Toc513155520)

[3.2.1 „Make or Buy“-Entscheidung 5](#_Toc513155521)

[3.2.2 Projektkosten 5](#_Toc513155522)

[3.2.3 Amortisationsdauer 5](#_Toc513155523)

[3.3 Nutzwertanalyse 5](#_Toc513155524)

[4 Entwurfsphase 6](#_Toc513155525)

[4.1 Entwurf der Benutzeroberfläche 6](#_Toc513155526)

[5 Implementierungsphase 7](#_Toc513155527)

[5.1 Implementierung der Datenstrukturen (Model) 7](#_Toc513155528)

[5.2 Implementierung der Benutzeroberfläche (View) 7](#_Toc513155529)

[5.3 Implementierung der Geschäftslogik (Controller) 7](#_Toc513155530)

[6 Abnahme- und Einführungsphase 8](#_Toc513155531)

[7 Fazit 9](#_Toc513155532)

[7.1 Soll-/Ist-Vergleich 9](#_Toc513155533)

[7.2 Lessons Learned 9](#_Toc513155534)

[7.3 Ausblick 9](#_Toc513155535)

[Literaturverzeichnis 10](#_Toc513155536)

[A Anhang i](#_Toc513155537)

[A.1 Detaillierte Zeitplanung i](#_Toc513155538)

[A.2 Verwendete Ressourcen ii](#_Toc513155539)

[A.3 Oberflächenentwurf iii](#_Toc513155540)

[A.4 Klassendiagramm iii](#_Toc513155541)

[A.5 Zustandssdiagramm iii](#_Toc513155542)

Abbildungsverzeichnis

[Abbildung 1: Entwurf der grafischen Oberfläche iv](file:///C:\Users\up40008\Desktop\Projektarbeit%202018\Dokumentation\Dokumentation%20Projekt%20Kiank%20-%20NMT.docx#_Toc513026172)

Tabellenverzeichnis

[Tabelle 1: Grobe Zeitplanung der Projektphasen 3](#_Toc513117375)

[Tabelle 2: Detaillierte zeitliche Aufschlüsselung der Phasen ii](#_Toc513117376)

Abkürzungsverzeichnis

|  |  |
| --- | --- |
| Begriff | Erläuterung |
| **bitGo\_GS** | **BIT**MARCK-**G**eschäftsprozesse **O**nline **G**eschäft**s**stelle  Abteilung in der die Anwendung eingesetzt werden soll |
| **BMT** | BITMARCK Technik GmbH |
| **CBW** | College Berufliche Weiterbildung GmbH |
| **CMS** | Content-Management-System: dt. Inhaltsverwaltungssystem |
| **GUI** | Graphical User Interface: grafische Benutzeroberfläche |
| **MVC** | Model-View-Controller: ist ein Architektur Model |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Einleitung

Im Rahmen einer Umschulung zum Fachinformatiker für Anwendungsentwicklung bei der CBW (College Berufliche Weiterbildung GmbH), absolvierte der Autor ein Praktikum bei der BITMARCK Technik GmbH in Hamburg und fertigte dort ebenfalls diese Dokumentation über die betriebliche Projektarbeit an.

Die Dokumentation ist mit Microsoft Word 2016 erstellt worden. Für die Erstellung der Präsentation wird Apples Keynote in der Version 8.0 verwendet.

Die Daten und Informationen in diesem Dokument sind Eigentum der BITMARCK Technik GmbH, eine Weitergabe an Dritte ist nicht gestattet.

* 1. Projektumfeld

Die BITMARCK Technik GmbH ist eine der Tochtergesellschaften BITMARCK Unternehmensgruppe und spaltete sich aus dem ehemaligen IT-Bereich der DAK-Gesundheit (Deutsche Angestellten Krankenkasse) ab. Die BITMARCK ist ein „… Full-Service-Dienstleister im IT-Markt der gesetzlichen Krankenversicherung und realisiert IT-Lösungen für die Betriebs- und Innungskrankenkassen sowie für die DAK-Gesundheit und weitere Ersatzkassen – 30.000 Mitarbeiter und 20 Millionen Versicherte in der GKV profitieren von den IT-Dienstleistungen der BITMARCK, 85 Prozent der Krankenkassen sind Kunden der Unternehmensgruppe.“ [[1]](#footnote-1)

* 1. Projektbegründung

Ein Produkt der BITMARCK ist die bitGo\_Suite. Sie ist eine Zusatzsoftware mit Anbindung an das Kernsystems BITMARCK\_21c|ng und besteht aus drei Komponenten:

Die Online-Geschäftsstelle bitGo\_GS[[2]](#footnote-2) ist der Anlaufpunkt für registrierte Versicherte.

Mit bitGo\_KV[[3]](#footnote-3) haben Krankenkassen die Möglichkeit, mit den Versicherten, die nicht in der Onlinekasse registriert sind, Schriftverkehr in Form von Formularen, Anträgen oder Umfragen zu führen.

Die bitGo\_App ist das jüngste Mitglied der bitGo\_Suite und bietet eine mobile Lösung mit Anbindung an das Kernsystem.

Die Komponenten der bitGo\_Suite liegen auf unterschiedlichen Servern. Krankenkassen beispielsweise arbeiten mit der Software, die auf einem Auslieferungsserver[[4]](#footnote-4) liegt. Zusätzlich wird jede Komponente mit Hilfe unterschiedlicher Microservices[[5]](#footnote-5) entwickelt. Die bitGo\_GS realisiert zum Beispiel Templates für die grafische Oberfläche mit dem Content-Management-System[[6]](#footnote-6) FirstSpirit.

Tritt während der Entwicklung ein Fehler auf Grund eines ausgefallenen Servers auf, lässt sich dieser aufgrund ungenauer Fehlerausgaben nicht exakt identifizieren. Mühsam wird jeder Server auf seine Verfügbarkeit hin überprüft. Ist die fehlende Verbindung gefunden, wird manuell ein Neustart ausgelöst.

* 1. Projektziel

Aufgabe ist es, eine Desktop-Applikation in der Programmiersprache Java zu entwickeln. In dieser sollen die Server und ihre Verfügbarkeiten aufgelistet sein. Bei einem Ausfall eines Servers, soll der Benutzer durch die Anwendung gewarnt werden. Die Verfügbarkeit von einem oder gleich aller Server soll der Benutzer während der Laufzeit, entweder manuell oder automatisch in bestimmten Zyklen, durch Anpingen[[7]](#footnote-7) ermitteln können. Das Ergebnis soll auf einer grafischen Oberfläche ausgegeben werden. Für die Anfragen notwendige Server- und Porteinträge sollen aus einer externen Quelle gelesen werden. Sie sollen von der Anwendung aus bearbeitet, gelöscht oder neu erstellt werden können.

* 1. Zielgruppe

Zielgruppe der Anwendung sind die Mitarbeiter des bitGo\_GS-Teams innerhalb der BITMARCK Technik GmbH.

1. Projektplanung
   1. Projektphasen

Für die Umsetzung des Projektes standen dem Autor 70 Stunden zur Verfügung. Diese wurden vor Projektbeginn grob in sechs Phasen aufgeteilt, die während der Softwareentwicklung durchlaufen werden. Die genaue Zeitplanung der einzelnen Phasen lassen sich der Tabelle 2 im Anhang A.1 entnehmen.

|  |  |
| --- | --- |
| Phase | Geplante Zeit |
| Analysephase | 2 h |
| Entwurfsphase | 11 h |
| Implementierungsphase | 36 h |
| Test und Korrektur | 6 h |
| Projektübergabe | 3 h |
| Dokumentation | 12 h |
| **Gesamt** | **70 h** |

Tabelle 1: Grobe Zeitplanung der Projektphasen

* 1. Ressourcenplanung

Die für das Projekt eingesetzten Ressourcen werden im Anhang A.2 Verwendete Ressourcen auf Seite iii aufgelistet. Damit sind sowohl Hard- und Softwareressourcen als auch das Personal gemeint. Bei der Auswahl der verwendeten Software wurde darauf geachtet, dass diese kostenfrei (z.B. als Open Source[[8]](#footnote-8)) zur Verfügung steht oder die BITMARCK Technik GmbH bereits Lizenzen für diese besitzt. Dadurch sollen anfallende Projektkosten möglichst gering gehalten werden.

* 1. Entwicklungsprozess

Bei der BITMARCK Technik GmbH wird nach dem agilen Vorgehensmodell Scrum[[9]](#footnote-9) Software entwickelt. Für das Abschlussprojekt, musste der Autor sich demnach nicht selbst für ein passendes Vorgehensmodell entscheiden.

Scrum verfolgt den Ansatz empirisch[[10]](#footnote-10), inkrementell[[11]](#footnote-11) und iterativ[[12]](#footnote-12) zu sein. Das bedeutet im groben, dass das langfristige Ziel (das Product Backlog[[13]](#footnote-13)) für die Fertigstellung des Projektes kontinuierlich verfeinert und verbessert wird. Genaue Anforderungen werden in sogenannten Sprints[[14]](#footnote-14) unterteilt. Da das Projekt einen relativ kurzen Zeitraum umfasst, wurden kleinere Sprints im Abstand einer Woche geplant. Das Projekt wurde somit auf zwei Wochen aufgeteilt.

In der ersten Woche wurden die Analysephase, die Entwurfsphase und ein Teil der Implementierungsphase durchlaufen.

In der Analysephase wurden der Ist-Zustand, der Soll-Zustand und die (daraus ergebene) Wirtschaftlichkeit ermittelt.

Weiter wurde in der Entwurfsphase ein Mockup[[15]](#footnote-15) der Anwendung erstellt. Der Funktionsumfang der Anwendung wurde in einem Klassendiagramm abgebildet (Siehe Anhang A.5: Klassendiagramm) und der Anwendungsablauf in einem Zustandsdiagramm (Siehe Anhang A.5: Aktivitätsdiagramm).

Im ersten Teil der Implementierungsphase wurde die Datenstruktur bzw. die Geschäftslogik realisiert. Es handelt sich hierbei um die sogenannten Model-Klassen[[16]](#footnote-16) der MVC-Softwarearchitektur[[17]](#footnote-17). Da es sich um eine testgetriebene Entwicklung handelt, wurde im Vorfeld für Klassen und Methoden JUnit-Tests[[18]](#footnote-18) geschrieben. Dadurch konnte sichergestellt werden, dass bei einer Veränderung des Quellcodes das Programmverhalten nicht unbeabsichtigt geändert wird.

1. Analysephase
   1. Ist-Analyse

Wie bereits im Abschnitt 1.2 (Projektbegründung) erwähnt wurde, ist das ...

* 1. Wirtschaftlichkeitsanalyse

Aufgrund der oben genannten Probleme, müssen folgende Fragen geklärt werden. Existiert auf dem Markt bereits ein Produkt, welches die spezifischen Anforderungen erfüllt und zusätzlich kosteneffizient ist oder es ist sinnvoller die Anforderungen, in Form einer eigen entwickelten Anwendung im Betrieb umzusetzen. In den folgenden Abschnitten wird der wirtschaftliche Vorteil einer Eigenentwicklung geklärt.

* + 1. „Make or Buy“-Entscheidung

Dieses Projekt ist zwar eine Anforderung der BMT für interne Zwecke, jedoch keine unternehmensspezifisches

Es handelt sich bei diesem Projekt um eine unternehmensspezifische Anforderung der BMT handelt. Alle Produkte der bitGo\_Suite

* + 1. Projektkosten
    2. Amortisationsdauer
  1. Nutzwertanalyse

... Pro und Kontra für Kauf oder selber entwickeln

1. Entwurfsphase
   1. Entwurf der Benutzeroberfläche
2. Implementierungsphase
   1. Implementierung der Datenstrukturen (Model)
   2. Implementierung der Benutzeroberfläche (View)
   3. Implementierung der Geschäftslogik (Controller)
3. Abnahme- und Einführungsphase
4. Fazit
   1. Soll-/Ist-Vergleich

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Soll | Ist | Differenz |
| Analysephase | 2 h | 4 h | +2 |
| Entwurfsphase | 11 h | 13 h | +2 |
| Implementierungsphase | 36 h | 36 h | 0 |
| Test und Korrektur | 6 h | 6 h | 0 |
| Projektübergabe | 3 h | 1 h | -2 |
| Dokumentation | 12 h | 10 h | -2 |
| **Gesamt** | **70 h** | **70 h** | **0 h** |

Tabelle 2: Soll-/Ist-Vergleich

* 1. Lessons Learned
  2. Ausblick

Literaturverzeichnis

#### UML 2.5 - Das umfassende Handbuch 2015

© Rheinwerk Verlag GmbH, Bonn 2015 – ISBN 978-8362-2977-7

1. Anhang
   1. Detaillierte Zeitplanung

|  |  |  |  |
| --- | --- | --- | --- |
| Analysephase |  |  | h |
| Ist-Zustandes ermitteln (Fachgespräch) |  | h |  |
| Projektplan | h |  |  |
|  | h |  |  |
|  |  | h |  |
|  |  | h |  |
| **Entwurfsphase** |  |  | **h** |
|  |  |  |  |
| **Entwicklungsphase** |  |  | **h** |
|  |  |  |  |
| **Projektübergabe** |  |  | **h** |
|  |  |  |  |
| **Gesamt** |  |  | **h** |

Tabelle 3: Detaillierte zeitliche Aufschlüsselung der Phasen

* 1. Verwendete Ressourcen

#### Hardware

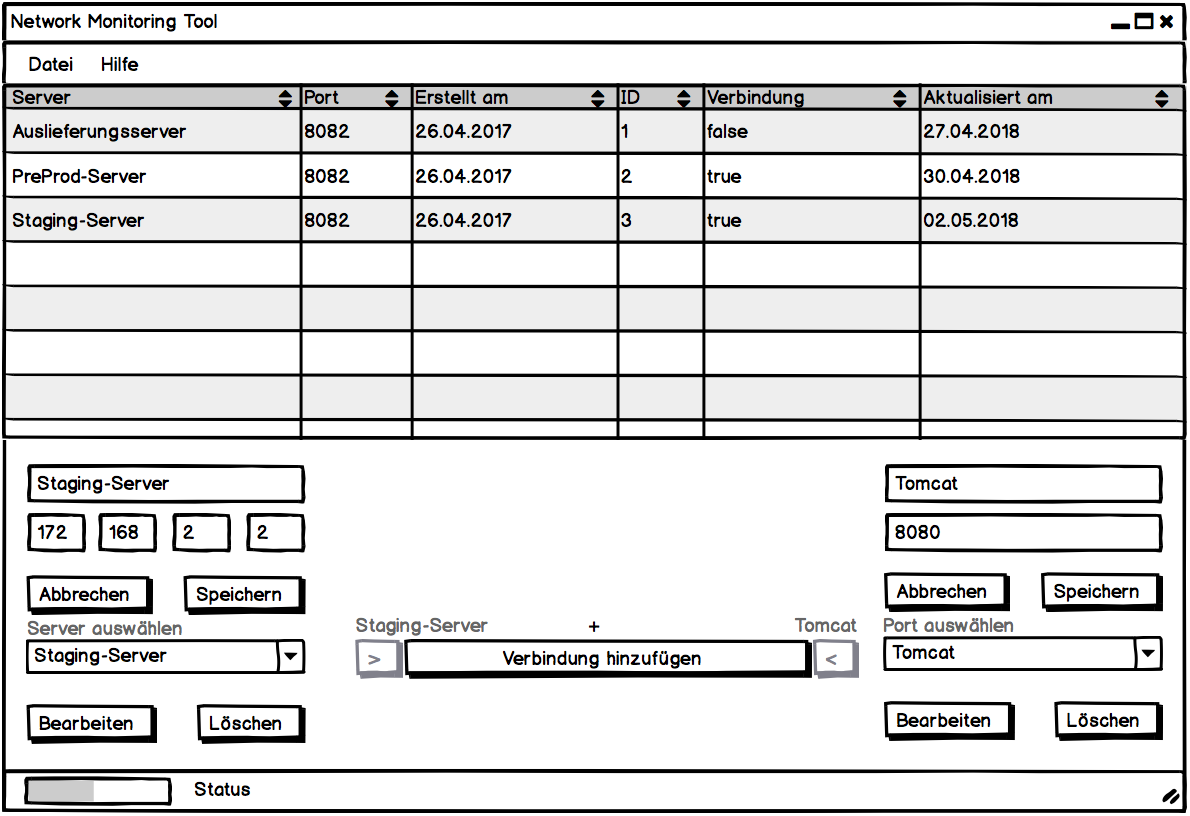
* Büroarbeitsplatz mit Fat-Client

#### Software

* Windows 10 – Betriebssystem
* Eclipse Oxygen – Entwicklungsumgebung Java
  + Import von Bibliotheken: JavaFX, JUnit 4, GSON
* Microsoft Office 365 – Bürosoftware
* Sourcetree (Git) – Verteilte Versionsverwaltung
* JavaFX Scene Builder 9.0.1 – Tool zum erstellen der GUI
* UMLet 14.2 Eclipse Plug-In – Tool zum erstellen von UML-Diagrammen
* Balsamiq.cloud– Webbasiertes Tool zum erstellen von Mockups

#### Personal

* Entwickler - Projektverantwortlicher und -ersteller
* Anwendungsentwickler – Review des Codes, Festlegung der Anforderungen und Abnahme des Projektes
  1. Oberflächenentwurf



* 1. Klassendiagramm
  2. Zustandssdiagramm

1. Vgl. <https://www.bitmarck.de> [↑](#footnote-ref-1)
2. GS=Geschäftsstelle [↑](#footnote-ref-2)
3. KV=Krankenversicherung [↑](#footnote-ref-3)
4. Hier liegt finale Version der Software, die an die Krankenkassen verteilt wird. [↑](#footnote-ref-4)
5. Ist ein unabhängiger Prozess einer komplexen Anwendungssoftware.

   Vgl. <https://de.wikipedia.org/wiki/Microservices> [↑](#footnote-ref-5)
6. „... ist eine Software zur gemeinschaftlichen Erstellung, Bearbeitung und Organisation von Inhalten...“

   Vgl. <https://de.wikipedia.org/wiki/Content-Management-System> [↑](#footnote-ref-6)
7. Senden von Testdaten an einen Server. [↑](#footnote-ref-7)
8. Offener, frei zugänglicher Quelltext der von Dritten eingesehen, geändert und genutzt werden kann.

   Vgl. <https://de.wikipedia.org/wiki/Open_Source> [↑](#footnote-ref-8)
9. Vgl. <https://de.wikipedia.org/wiki/Scrum> [↑](#footnote-ref-9)
10. Bedeutung: erfahrungsgemäß [↑](#footnote-ref-10)
11. Bedeutung: regelmäßig [↑](#footnote-ref-11)
12. Bedeutung: wiederholend [↑](#footnote-ref-12)
13. „...ist eine geordnete Auflistung der Anforderungen an das Produkt.“

    Vgl. <https://de.wikipedia.org/wiki/Scrum#Product_Backlog> [↑](#footnote-ref-13)
14. „...ein Arbeitsabschnitt, in dem ein Teil einer Produktfunktionalität implementiert wird.“

    Vgl. <https://de.wikipedia.org/wiki/Scrum#Sprint> [↑](#footnote-ref-14)
15. „dt. Vorführmodell“, Um die Funktionalität eines kompletten Produktes zu präsentieren.

    Vgl. <https://de.wikipedia.org/wiki/Vorführmodell> [↑](#footnote-ref-15)
16. Vgl. <https://de.wikipedia.org/wiki/Model_View_Controller#Modell_(model)> [↑](#footnote-ref-16)
17. Model-View-Controller

    Vgl. <https://de.wikipedia.org/wiki/Model_View_Controller> [↑](#footnote-ref-17)
18. „... ist ein Framework zum testen von Java-Programmen,...“

    Vgl. <https://de.wikipedia.org/wiki/JUnit> [↑](#footnote-ref-18)