Pflichtenheft

| | |
|---|---|
| Projektbezeichnung | Vier gewinnt |
| Projektleiter | Christian Chimani!! |
| Erstellt am | 10.03.2022 |
| Letzte Änderung am | 17.03.2022 |
| Status | in Bearbeitung |
| Aktuelle Version | 1.2 |

# Änderungsverlauf

| Nr. | Datum | Version | Geänderte Kapitel | Art der Änderung | Autor | Status |
|---|---|---|---|---|---|---|
| 1 | 10.03.2022 | 1.0 | Alle | Erstellung | Wittner Michael | - |
| 2 | 10.03.2022 | 1.1 | Alle | Korrekturen | 3AHITN | - |
| 3 | 17.03 | 1.2 | 3 | Korrekturen | cchimani | - |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | criedler<br>emiklaut | |
| 4 | | | | | | |
| 5 | | | | | | |

fg fertiggestellt

iB in Bearbeitung

ab abgebrochen

Pr Prüfung

paus pausiert

# Inhalt

# 1 Zielbestimmung

Im vorliegenden Pflichtenheft werden alle funktionalen und nichtfunktionalen Anforderungen an die Uhr beschrieben.

Es muss eine einfach zu bedienende Anwendung erstellt werden, welche eine Smart Watch bedient. Die Implementierung hat in den eingeteilten Gruppen zu erfolgen. Um eine einfache Projektorganisation zu gewährleisten, ist ein Softwareverwaltungssystem einzusetzen.

## 1.1 Musskriterien

- Es muss die Uhrzeit ausgegeben werden
- Zur vollen Stunde kommt eine Benachrichtigung
- Aktuelles Datum
- Kalenderwoche
- Stoppuhr
- Night/Day Mode
  - 12h/24h Zeitmodus umstellen
  - Zeitformat: Analog / Digital / Binary

## 1.2 Wunschkriterien

- Luftfeuchtigkeit
- Temperatur
- Geburtstag eintragen
- Wettersymbol (z.B. ☺≋☂)

# 2 Allgemeines

## 2.1 Ausgangssituation

Im letzten Projekt wurde ein Spiel nach dem MVC-Prinzip aufgebaut. Es wurde die Aufteilung in verschieden Klassen vorgegeben.

Innerhalb der Model-Klassen müssen alle für die Uhrlogik notwendigen Teile enthalten sein.

Die View-Klassen dienen zur Darstellung der Daten aus den Modellklassen, es darf keine direkte Beziehung zwischen Model und View-Klassen geben. Als Schnittstelle zwischen diesen dient die Controller Klasse. Diese Klasse ist für die Ausführung der Daten von der Uhr verantwortlich.

Nach Aktualisierung der Daten durch das Modell müssen diese mit Hilfe der View-Klassen dargestellt werden.

Das vorliegende Projekt muss nach dem gleichen Prinzip aufgebaut werden.

## 2.2 Team

In der nachfolgenden Tabelle sind die Daten zu den Teammitgliedern einzutragen. Seite 1 ist zu ändern, das Dokument muss in das Git-Repository geladen werden.

| Rolle(n) | Name | Git-Account | E-Mail |
|---|---|---|---|
| Projektleitung | Christian Chimani | cchimani | cchimani@htl-steyr.ac.at |
| Entwicklung | Constantin Riedler | criedler | criedler@htl-steyr.ac.at |
| Entwicklung | Elias Miklautsch | emiklaut | emiklaut@htl-steyr.ac.at |
| | | | |

# 3 Funktionale Anforderungen

- Es muss die Uhrzeit ausgegeben werden
- Zur vollen Stunde kommt eine Benachrichtigung

- Aktuelles Datum
- Kalenderwoche
- Stoppuhr
- Night/Day Mode
- 12h/24h Zeitmodus umstellen
- Zeitformat in Binary umstellen

## 3.1 Konsolenanwendung

Nur zum Testen

## 3.2 Grafische Anwendung

- Beim Starten der Applikation wird ein Fenster aufgerufen welches eine Uhr mit den zuletzt eingestellten Settings öffnet.
- Die Uhrzeit kann über ein Menü eingestellt werden
- An der Uhr können folgende Einstellungen vorgenommen werden:
    - Digitaluhr
        - 24 Stundenformat, 12 Stundenformat
        - Schriftart / Schriftgröße
    - Binary Uhr (Beispielbild)
        - Farbe der Leds muss einstellbar sein
    - AnalogUhr (Beispielbild)
        - Sekundenzeiger ein-/ausschaltbar
        - Datum sichtbar
- Stoppuhrmodus
    - Über Einstellungen wählbar
    - Mit einem Start Button kann die Stoppuhr dann gestartet werden
    - Neuer Button Klick stoppt die Uhr - Anzeige bleibt erhalten!
- Timermodus:
    - Über Einstellungen wählbar
    - Die Zeit (Stunden, Minuten, Sekunden) muss konfigurierbar sein
    - Der Timer muss mit einem Button gestartet werden können
- Die Benachrichtigung der vollen Stunde erscheint mit einem Pop-up Fenster.
- Je nach Zeit wird der Hintergrund der Uhr auf Nacht oder Tag gewechselt. (z.B. Grün für Tag, Dunkelblau für Nacht).

# 4 Nichtfunktionale Anforderungen

## 4.1 Programmiersprache / Programmierumgebung

Die Anwendung muss in der Programmiersprache Java umgesetzt werden. Für die grafische Anwendung ist JavaFX einzusetzen.

Als JavaFX Version muss die Version verwendet werden, welche am Schulserver unter

//acdc/software/Updates/Microsoft/Windows10/Applications/Java

zu finden ist.

Als Entwicklungsumgebung muss IntelliJ der Firma JetBrains (https://www.jetbrains.com/de-de/idea/) eingesetzt werden.

## 4.2 Diagramme

Für die Erstellung von Diagrammen muss http://www.draw.io oder https://app.genmymodel.com verwendet werden. Es ist darauf zu achten die Diagramme so in der Versionsverwaltungssoftware zu speichern, dass diese weiter bearbeitet werden können.

Änderungen in Diagrammen müssen dokumentiert (einschließlich Begründung) werden.

## 4.3 Programmierrichtlinien / Dokumentation

Bei der Programmierung ist auf die Einhaltung der Programmierrichtlinien der HTL Steyr zu achten – siehe

http://www.htl-steyr.ac.at/intern/wiki/doku.php?id=el-it:fsst:softwareentwicklung:programmierrichtlinien

Die Inline-Dokumentation hat den Richtlinien zu entsprechen, welche im obigen Dokument beschrieben werden.

Die „Readme.md“ Datei muss alle für das Projekt notwendigen Informationen beinhalten.

Der Arbeitsfortschritt muss in Trello dokumentiert werden. Für jedes Teammitglied muss ein Label erstellt werden. Jeder Card muss ein Label zugeordnet werden, welcher zeigt welches Team-Mitglied für die Umsetzung verantwortlich ist.

## 4.4 Softwareversionierung

Für die Teamarbeit sowie für die Softwareversionierung ist GitHub zu verwenden. Am Ende jedes Unterrichtsblocks müssen Änderungen in das Repository hochgeladen werden. Dabei ist darauf zu achten, dass im main-Branch immer ein lauffähiges Programm existieren muss.

Die Entwicklung der einzelnen Teilbereiche hat in eigenen Branches zu erfolgen. Jedes Teammitglied muss in seinem Branch arbeiten.

Alle Dateien welche für das Projekt notwendig sind müssen in GitHub enthalten sein.

## 4.5 Arbeitsbericht

Im Repository muss für jedes Teammitglied eine „.md“-Datei für einen Arbeitsbericht erstellt werden. Im Arbeitsbericht muss für jede Unterrichtsstunde von jedem Teammitglied dokumentiert werden, woran gearbeitet wurde, welche Probleme aufgetreten sind, und welche Teile fertig implementiert wurden.

# 5 Rahmenbedingungen

Vor der Implementierung ist ein Klassendiagramm zu erstellen welches alle notwendigen Klassen für die Konsolen- und für die grafische Anwendung beinhaltet.

## 5.1 Zeitplan

Tag 1 (2h) 10.03: Ideensammlung Pflichtenheft

Tag 2 (2h) 17.03: Pflichtenheft

Tag 3 (2h) 18.03: Kanban

Tag 4 (2h) 24.03: Umsetzung/Test View Klassen Konsolenanwendung

Tag 5 (2h) 25.03: Umsetzung/Test View-Klassen GUI Anwendung

Tag 6 (2h) 01.04: Umsetzung/Test View-Klassen GUI Anwendung

Tag 7 (2h) 07.04: Umsetzung/Test View-Klassen GUI Anwendung

Tag 8 (2h) 08.04: Umsetzung/Test View-Klassen GUI Anwendung

# 6 Abgabebedingungen - Notengebung

Die fertige (Teil-) Software ist am Abgabedatum in GitHub abzulegen.

Im „Main“-Branch ist

- die fertige Konsolenanwendung mit dem Tag „Console“ zu benennen
- die fertige GUI-Anwendung mit dem Tag „Gui“ zu benennen

Nicht lauffähige Projekte werden negativ beurteilt.

Während der Zeit des Projektes werden Aufzeichnungen zu folgenden Bereichen erstellt:

- Einhaltung der zeitlichen Vorgaben
- Arbeiten mit Versionskontrollsoftware
- Kommunikation innerhalb des Teams
- Dokumentation der Fortschritte

https://herrenuhren-xxl.de/wp-content/uploads/digitaluhren.jpg