1. 7.       Iterationenplan (Timeboxes)
   1. Überblick

Kriteien:

Ranking:

UseCases werden geordnet nach ihrer Bewertung. Die drei höchstbewerteten werden in den drei

ersten Timeboxes realisiert. Die Berwertung geschieht nach folgenden Kriterien:

1. Risiko
   * Komplexität des UseCses
   * Anforderungen (inkl. an die Usability): haben/bekommen wir die richtigen.Anforderungen?  wie stabil sind sie? haben wir Ansprechpartner?
   * Team KnowHow: der dazu notwendigen Technologie, und des Domänenverständnisses
   * Technologie: ist die einzusetzendeTechnologie gut definiert, stabil, erprobt, einfach?
   * politisch
2. Architekturrelevanz gibt die Realisierung des UseCases wichtige Einblicke in die SW-Architektur des Systems? Hilft er bei den wichtigsten Entscheidungen zur Architektur?
3. Benutzerrelevanz ist der Benutzer an einer frühen Realiserung dieses UseCases interessiert?
4. Lern-und Entwicklungsrelevanz ist der UseCase geeignet, um mangelndes teaminternes KnowHow bzgl. GUI-Entwurf, Klassenentwurf, Datenbankanbindung etc. aufzubauen?
5. UseCase-Ranking:

|  |  |
| --- | --- |
| 10... | sehr relevant |
| 0... | nicht relevant |

HIER KOMMT RANKING HIN

* 1. 1. Timebox
     1. Benutzungsfall/fälle (UseCase(s))

Basisabläufe:

* Check-In
* Check-Out
* Reservierung buchen
  + 1. Architektur
* Datenbankmodellierung basierend auf Domänenmodell
* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING
  + 1. Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Relationales Datenbankmodell

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

* + 1. Abhängigkeiten
* Umzusetzende Use cases sind komplett definiert
* Domänenmodell ist vorhanden
* Datenbankserver ist vorhanden und konfiguriert
* Testumgebung/en vorhanden
  1. 7.3     2. Timebox
     1. 7.3.1         Benutzungsfall/fälle (UseCase(s))

Basisabläufe:

* Tagesabschluss
* Rechnung erstellen
* Zwischenrechnung erstellen
* Rechnung legen
  + 1. Architektur
* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING
  + 1. Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Keine

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

* + 2. Abhängigkeiten
* Fertigstellung der ersten Timebox
* Umzusetzende Use cases sind komplett definiert
  1. 7.4     3. Timebox
     1. 7.4.1         Benutzungsfall/fälle (UseCase(s))

Basisablauf:

* Belegungsvorschau

Alternativabläufe:

* Check-In
* Check-Out
* Reservierung buchen
  + 1. Architektur
* Datenbankanbindung mittels Hibernate
* GUI basierend auf Java SWING
  + 1. Deliverables

Code:  
Java Code nach teaminternen Konventionen

Manual:  
Javadoc

Spezifikationen:  
Keine

Testdaten:  
Werden manuell in die Datenbank übertragen.

Testergebnisse:  
Basisablauf der definierten Use cases funktioniert fehlerfrei.

* + 2. Abhängigkeiten
* Fertigstellung der zweiten Timebox
* Umzusetzende Use cases sind komplett definiert