Für internen Gebrauch

Yatzy

autor(en) : Bosshard Fabrice

version : 1.0

status : Final

quelle : Atos

dokumentendatum : 21 November 2018­

­

­

anzahl der seiten : 4

owner : Fabrice-Ronny Bosshard

**Änderungshistorie**

| Version | Datum | Beschreibung | Autor(en) |
| --- | --- | --- | --- |
| 0.1 | 21.11.2018 | Initiale Dokumentationsstruktur erstellt | Fabrice Bosshard |
| 0.2 | 22.11.2018 | Informieren abgeschlossen | Fabrice Bosshard |
| 0.3 | 22.11.2018 | Planen abgeschlossen | Fabrice Bosshard |
| 0.4 | 22.11.2018 | Entscheiden abgeschlossen | Fabrice Bosshard |
| 0.5 | 27.11.2018 | Realisieren abgeschlossen | Fabrice Bosshard |
| 0.6 | 27.11.2018 | Kontrollieren abgeschlossen | Fabrice Bosshard |
| 0.7 | 28.11.2018 | Auswerten abgeschlossen | Fabrice Bosshard |
| 0.8 | 28.11.2018 | Finale Version abgeschlossen | Fabrice Bosshard |

Tabelle 1: Änderungshistorie

© Copyright 2018, Atos AG Alle Rechte vorbehalten. Reproduktion von Teilen oder dem Gesamten ist ohne vorherige schriftliche Genehmigung des Urhebers untersagt. Für Fragen oder Anmerkungen zu diesem Dokument wenden Sie sich bitte an Atos, 078 620 00 35.

­­­Inhalt

**Teil 1: Umfang und Ablauf**

1 Aufgabenstellung 4

1.1 Projekt 4

1.2 Ausgangslage 4

1.3 Detaillierte Aufgabenstellung 4

1.4 Management Summary 4

1.5 Technologie-Stack 4

1.6 Vorkenntnisse 4

1.7 Vorarbeiten 4

2 Projektorganisation 5

2.1 Beteiligte Personen 5

2.2 Projektmanagementmethode 5

2.3 Backup-Konzept 5

3 Zeitplanung 6

3.1 Meilensteine 6

3.2 Massnahmen bei Verzug 6

4 Gantt-Diagramm 7

5 Arbeitsprotokoll 8

5.1 Mittwoch 21. November 2018 (3h) 8

5.2 Donnerstag 22. November 2018 9

5.3 Freitag 23. November 2018 10

5.4 Dienstag 27. November 2018 (3h) 11

6 Kurzfassung 12

6.1 Ausgangslage 12

6.2 Umsetzung 12

6.3 Ergebnis 12

7 Informieren 13

8 Planen 14

9 Entscheiden 15

10 Realisieren 16

11 Kontrollieren 17

12 Auswerten 18

13 Glossar 19

14 Verzeichnisse 20

14.1 Quellenverzeichnis 20

14.2 Abbildungsverzeichnis 20

14.3 Tabellenverzeichnis 21

15 Aufgabenstellung 22

16 Programmcode 23

# Aufgabenstellung

Die Aufgabenstellung wird vom Auftraggeber festgelegt und ist anbei dokumentiert.

## Projekt

Projekttitel: Yatzy

Im Auftrag von: Torben Dziuk

Auftragnehmer: Fabrice-Ronny Bosshard

Starttermin: 21. November 2018 14:00

Abgabetermin: 28. November 2018 13:00

Geplanter Projektaufwand: Drei Arbeitstage à 8 Stunden (24h+ 6h)

Erstellung einer simplen Applikation, mit Frontend, Backend und Anbindung an eine Datenbank.

## Ausgangslage

Damit die Lehrlinge in der Atos genügend auf die Individuelle Praktische Arbeit (IPA) vorbereitet sind, leitet Torben Dziuk jeden Mittwoch einen Kurs im kleinen Rahmen. Nun geht es darum eine Probe-IPA zu schreiben, damit wir aus unseren Fehlern lernen können und uns stetig verbessern.

Als Aufgabenstellung wurde die Aufgabe aus dem Informatik-Schweizermeisterschaften des Jahres 2017 genommen.

Das Projekt wird von Grund auf aufgebaut und implementiert. Der Auftragnehmer muss neben den Standard .NET-Kenntnissen auch solide Kenntnisse in Entity Framework und WPF (Windows Presentation Foundation) haben.

Die Struktur der Dokumentation soll bereits als Vorlage existieren, damit bei Projektbeginn direkt mit dem Schreiben angefangen werden kann.

## Detaillierte Aufgabenstellung

Im Anhang finden sie die detaillierte Aufgabenstellung der ICT-Skills Schweiz. Sie beschreibt die funktionalen Anforderungen an die Applikation und gibt durch Wire Frames vor, wie das Layout der Benutzeroberfläche aussehen muss.

## Management Summary

Kurze Repräsentation der Aufgabenstellung von ICT-Skills Schweiz.

Yatzy ist ein Würfelspiel für Klein bis Gross. Das Ziel dieses Spieles ist, die meisten Punkte mit den verschiedenen vorgegebenen Kombinationen zu sammeln. Sobald alle Kombinationen abgedeckt sind, gewinnt der Spieler mit der höchsten Punktzahl.

## ****Technologie-Stack****

IDE: Visual Studio 2015 (mit ReSharper)

Database: MSSQL 2014 mit Management Studio

Front- / Backend: C# 5.0 mit Windows Presentation Foundation (WPF)

Versionsverwaltung: Git mit TortoiseGit

Programmarchitektur wird nach MVVM (Model View ViewModel) umgesetzt.

## Vorkenntnisse

In diesem Abschnitt werden die bereits erlangten Kenntnisse aufgelistet.

Projektkenntnisse:

* C# (2 Jahre)
* WPF (2 Jahre)
* Visual Studio (2 Jahre)
* Angular *TypeScript* (1 Jahr)
* Git (2 Jahre)
* CI (1 Jahr)
* DI (1 Jahr)

## Vorarbeiten

In der IPA muss eine vorgegebene Arbeit umgesetzt und dokumentiert werden. Oftmals sprengen alle Tätigkeiten in dieser Arbeit den Zeitrahmen und deshalb werden gewisse Arbeiten bereits zuvor erledigt. In diesem Kapitel werden alle Vorbereitungen für die Probe-IPA deklariert.

* Initiale Dokumentationsstruktur aufbauen nach IPERKA.
  + Struktur
  + Layout
  + Verzeichnisse

# Projektorganisation

Dieses Kapitel gibt einen Überblick über den Aufbau der Organisation, informiert über die beteiligten Personen und zeigt die Vorgehensweise im Allgemeinen.

## Beteiligte Personen

|  |  |  |  |
| --- | --- | --- | --- |
| **Person** | **Rolle** | **Verantwortung** | **Kontakt** |
| Torben Dziuk | Auftraggeber & Verantwortliche Fachkraft | Aufgabe definieren & Unterstützung Kandidat | torben.dziuk@atos.net |
| Fabrice Bosshard | Kandidat | Umsetzung der Arbeit | fabrice.bosshard.external@atos.net  fabrice.bosshard@siemens.com |

Tabelle 2: Beteiligte Personen

## Projektmanagementmethode

Im Betrieb arbeiten wir mit der Projektmanagementmethode Scrum, welche sich sehr gut für die agile Prozessentwicklung eignet. Bei Scrum werden die verschiedenen Arbeitsschritte in Zyklen (Sprints) abgehandelt, was das Anpassen von Vorgaben zwischen den Zyklen und die transparente Kommunikation innerhalb des Entwicklerteams ermöglicht.

Scrum ist jedoch für eine IPA nicht gut geeignet, da der Mehraufwand für die Formalitäten (Daily, Retro etc.) zu hoch wäre.

Ich habe mich deshalb für die Projektmanagementmethode IPERKA (**I**nformieren, **P**lanen, **E**ntscheiden, **R**ealisieren, **K**ontrollieren, **A**uswerten) entschieden, da ich damit schon viel Erfahrungen in der Schule sammeln konnte. Diese klassische Projektmanagementmethode eignet sich für die IPA perfekt, da der Ablauf respektive die Struktur mit den Anforderungen der Arbeit übereinstimmt.

## Backup-Konzept

Bei der Umsetzung der Arbeit ist es essentiell, dass man stets auf eine valide Version der Arbeit zurückgreifen kann. Wenn man zum Beispiel einen groben Fehler macht und die ganze Applikation nicht mehr zum Laufen kriegt, ist es wichtig, dass man nicht von Neuem beginnen muss, sondern auf den Stand vor dem Fehler zurückgreifen kann.

Die Dokumentation und der Quellcode werden in ein Git-Repository geladen und sind somit zentral in einer Cloud gespeichert. Dazu wird das externe Tool GitHub verwendet. GitHub macht bei sich selber immer wieder Backups und so muss ich mich nicht um ein Backup-System kümmern. Das Repository muss stets aktualisiert werden, damit möglichst immer der neuste Stand verfügbar ist. Um dies zu ermöglichen, sollte für jede neue Funktionalität ein Commit erstellt werden, welcher den neuen Stand darstellt (Daumenregel: 2-4 Mal am Tag).

Als weitere Absicherung wird ein USB-Stick genutzt, auf welchem mindestens ein Mal am Tag ein komplettes Backup erstellt wird. Im Ernstfall würde man als erstes versuchen auf das Git-Repository zurückzugreifen, doch wenn dies ebenfalls fehlschlagen sollte, gibt es noch eine Sicherung auf dem Stick.

# Zeitplanung

In diesem Kapitel werden die verschiedenen Aufwände geschätzt und aufgelistet. Die Zeitplanung dient als Leitfaden für den Projektablauf und hilft bei der Selbstkontrolle.

## Meilensteine

Meilensteine sind wichtige Teilpunkte im Projektverlauf. Sie werden als Prüfpunkte verwendet und wirken sich positiv auf die Qualitätssicherung aus. Man kann somit die erledigten Ergebnisse mit den erwarteten Vorgaben abgleichen.

|  |  |  |
| --- | --- | --- |
| Meilenstein | Erfüllungskriterien | Datum |
| Projektbeginn | Kick-off der Probe IPA | 21.11.2018 14:00 Uhr |
| Informieren | Auftragsanalyse durchgeführt und dokumentiert | 22.11.2018 10:00 Uhr |
| Planen | Anwendungsfälle, Architektur, Testfälle planen | 22.11.2018 15:00 Uhr |
| Entscheidung | Tools / Framework und Architektur-Entscheidungen definieren | 22.11.2018 17:30 Uhr |
| Realisieren | Implementation der Arbeit & Arbeit dokumentiert | 27.11.2018 15:00 Uhr |
| Kontrollieren | Die zuvor definierten Kontrollkriterien überprüft und Fazit daraus gezogen | 27.11.2018 17:30 Uhr |
| Auswerten | Reflexion der Arbeit und des Arbeitsverhalten erstellt & Kurzfassung geschrieben | 28.11.2018 11:00 Uhr |
| Projektabschluss | Projektarbeit abgeschlossen | 28.11.2018 13:00 Uhr |

Tabelle 3: Meilensteine

## Massnahmen um Verzug zu verhindern

Ein Zeitplan oder Ähnliches ist ein Leitfaden den es unbedingt braucht. Jedoch wird es in praktisch keinem Projekt gelingen, sich stets an den Plan zu halten. Man muss vorbereitet sein auf gewisse Abweichung, denn diese passieren überall. Deshalb wird in dieser Arbeit für jeden Tag die letzte Stunde als „offen“ geplant. Das heisst ich habe immer eine Stunde pro Tag, welche für nichts eingeplant wurde und welche ich nutzen kann, falls ich mit einer geplanten Arbeit etwas länger brauche als geplant.

## Massnahmen bei Verzug

Sobald ein Meilenstein oder eine Tätigkeit nicht zur definierten Zeit erledigt werden konnte, muss eine Massnahme ergriffen werden, damit weitere wichtige Tätigkeiten nicht in Gefahr geraten. Dazu würden die kleineren Teil-Tätigkeiten, welche nicht grösseren Einfluss auf den Fertigstellungsgrad haben, ignoriert und als optional für nach der Fertigstellung markiert werden.

Es werden jedoch nur Teil-Tätigkeiten aus der Realisierungs-Phase weggenommen, damit der allgemeine Projektablauf nicht manipuliert wird. Das heisst, wenn eine Massnahme ergriffen werden müsste, würde als Erstes die letzte Funktionalität laut Aufgabenstellung (siehe Anhang) rausfallen.

# Gantt-Diagramm



Abbildung 1: Gantt-Diagramm

# Arbeitsprotokoll

Das Arbeitsprotokoll dient zur Repräsentation des Tagesablaufs und der Fortschrittsbelegung. An jedem Arbeitstag wird das Arbeitsprotokoll erweitert und gepflegt. Es wird für jeden Tag mindestens eine Stunde für das Protokoll eingerechnet, damit die Qualität des Produkts sichergestellt werden kann. Es zeigt die geplanten Aufgaben pro Tag und stellt dar, ob es von der Tagesplanung Abweichungen gegeben hat. Zudem enthält das Protokoll jeweils eine Reflexion für jeden Tag, bei der man auf Erfolge, Misserfolge und den daraus zu schliessenden Massnahmen achtet.

## Mittwoch 21. November 2018 (3h)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag 1 (21.11.2018) | | | | |
| Phase: Informieren | | | | |
| Tätigkeit | **Soll-Zeit (Stunden)** | **Ist-Zeit (Stunden)** | **Bemerkung** |
| Dokumentationsstruktur | 0.5 | 0.5 |  |
| Projektorganisation definieren | 0.5 | 0.5 |  |
| Meilensteine definieren + Massnahmen | 0.5 | 0.5 |  |
| Gantt-Diagramm | 1 | 1 |  |
| Arbeitsjournal | 0.5 | 0.5 |  |
| Total | 3 Stunden | 3 Stunden |  |

Tabelle 4: Arbeitsprotokoll 21.11.2018

Heute habe ich mit dem Projekt gestartet. Als Erstes hatte ich die Aufgabe mit Torben Dziuk besprochen und wir haben kleinere Verständnisfragen zum Inhalt geklärt. Danach machte ich mich gleich daran die Dokumentationsstruktur aufzubauen, welche ich teilweise aus der bereits erstellen Probe-IPA aus dem Modul 226 übernommen habe. Den Rest der Zeit habe ich noch kleinere Tätigkeiten erledigt, welche bereits zur Phase „Informieren“ gehören.

### Erfolge

Ich konnte die Struktur des Dokuments fast 1 zu 1 von der Probe-IPA übernehmen. Ich merkte heute den enormen Mehrwert, einer guten Vorbereitung. Aus diesem Grund geling es mir recht schnell die geplanten Tätigkeiten abzuhandeln, obwohl ich relativ müde und deshalb leistungsineffizient war.

### Misserfolge

Ich finde es etwas Schade, dass wir Mitten im Tag mit dem Projekt gestartet haben. Die Anfangsphase (Informieren) ist sehr wichtig und da sollte man mit dem Kopf 100% dabei sein. Gegen Ende des Tages ist man jedoch meistens sehr angeschlagen und es fiel mir deshalb ein bisschen schwer die Zeitplanung zu erstellen. Ausserdem musste ich, sobald ich mal in der Arbeit drin war, direkt aufhören und die Arbeit am nächsten Tag weiterführen. Dies stört den Arbeitsfluss und verschlechtert meine Effizienz.

### Fazit

Ich konnte alle geplanten Tätigkeiten erledigen und bin vollständig bereit für den nächsten Tag. Durch die bereits durchgelaufene Probe-IPA habe ich viele Kenntnisse erworben, welche mir helfen effizienter und qualitativ besser zu arbeiten. Diese konnte ich schon heute einsetzen.

## Donnerstag 22. November 2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag 1 (22.11.2018) | | | | |
| Phase: Informieren, Planen & Entscheiden | | | | |
| Tätigkeit | **Soll-Zeit (Stunden)** | **Ist-Zeit (Stunden)** | **Bemerkung** |
| Ist-/Sollanalyse | 1 | 1 |  |
| Systemübersicht beschreiben | 0.5 | 0.5 |  |
| Use Cases erstellen | 1 | 1 |  |
| Testkonzept erstellen | 1 | 1 |  |
| Software-Architektur beschreiben | 0.5 | 1 |  |
| Datenbank-Architektur beschreiben | 0.5 | 0.5 |  |
| Tools und Framework entscheide | 1 | 1 |  |
| Datenbank-Implementation entscheiden | 0.5 | 0.5 |  |
| Arbeitsjournal schreiben | 1 | 1 |  |
| Total | 7 Stunden | 7.5 Stunden |  |

Tabelle 5: Arbeitsprotokoll 22.11.2018

Heute war sehr wahrscheinlich der strengste Tag von den Aufgaben her. Ich musste die Planung und die Entscheidung umsetzen (Informieren war am Morgen fertig), damit ich am Freitag und Dienstag 2 volle Tage für die Implementation habe. Ich merke nun am Ende des Tages, dass mein Kopf komplett ausgelastet ist und jetzt Feierabend braucht.

### Erfolge

Ich war sehr überrascht von der Leistung die ich erbringen konnte. Ich war zu Beginn des Tages äusserst müde, da ich wirklich schlecht geschlafen habe. Jedoch konnte ich trotzdem alle Arbeiten des Tages abschliessen und konnte auch recht gut komplexere Probleme angehen. Ich hatte fast keine Mühe mit den Tätigkeiten und musste auch keine Hilfe beanspruchen.

### Fazit

Es ist sehr anspruchsvoll den ganzen Tag voll konzentriert zu Planen und zu Dokumentieren. Mir fällt das viel schwerer als einfach zu programmieren. Ich habe jedoch gemerkt, wenn ich am Anfang des Tages eine Art Checkliste mache, bekomme ich fast mehr Motivation, denn ich kann die Dinge immer Abhacken und muss mich auch nicht immer um eine Struktur kümmern, sondern ich erstelle einmal eine Tages Planung und gehe danach durch die Arbeit wie ein Zombie. So arbeite ich am besten denke ich.

## Freitag 23. November 2018

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag 1 (23.11.2018) | | | | |
| Phase: Realisieren | | | | |
| Tätigkeit | **Soll-Zeit (Stunden)** | **Ist-Zeit (Stunden)** | **Bemerkung** |
| Umgebung aufsetzen | 0.5 | 0.5 |  |
| Datenbankstruktur erstellen & verknüpfen | 1 | 1 |  |
| Applikationsstruktur erstellen (Views, ViewModel & Model) | 3 | 3 |  |
| Business-Logik mit View und DatabaseClient verknüpfen | 3 | 3 |  |
| Arbeitsjournal | 1 | 1 |  |
| Total | 8.5 Stunden | 8.5 Stunden |  |

Tabelle 6: Arbeitsprotokoll 23.11.2018

### Erfolge

Heute habe ich den ganzen Tag implementiert. Ich habe mir an dem Tag vorgenommen, die ganze Funktionalität und die Benutzeroberfläche zu implementieren. Ich konnte eigentlich die ganze Applikation an diesem Tag entwickeln und kam vor allem schnell mit der Logik-Implementation voran.

### Misserfolge

Obwohl ich mit der Business-Logik sehr schnell vorankam, hatte ich grössere Mühe bei der Erstellung der Benutzeroberflächen. Ich hatte viel länger als erwartet und musste deshalb Zeit aufopfern, welche ich eigentlich nutzen wollte, um die Qualität des Codes zu verbessern. Nun sieht es so aus, dass die Qualität des Codes (Vor allem bezüglich Clean Code Prinzipien) nicht wirklich akzeptable ist und ich diese eigentlich noch überarbeiten möchte. Die Realisationsphase ist nun jedoch vorbei und ich werde kaum noch genügend Zeit haben, um noch Änderungen zu machen.

### Massnahmen

In diesem Projekt wäre es jetzt zu spät, aber ich werde auf jeden Fall, für die richtige IPA, noch ein bisschen lernen müssen. Ich werde die WPF - Kontrollelemente genauer anschauen und vor allem ihre Best Practices versuchen umzusetzen. Zudem muss ich auch noch die Best Practice der MVVM-Architektur kennenlernen. Ich weiss wie MVVM funktioniert, habe jedoch Ab und An Probleme diese nach den optimalen Richtlinien umzusetzen und dann gibt es manchmal eine Bastelarbeit in meinem Code.

### Fazit

Ich habe gespaltene Gefühle zu diesem Tag. Ich bin einerseits froh, dass ich die Funktionalität der Applikation fertig entwickeln konnte, bin jedoch auch ein wenig enttäuscht, dass durch den Mehraufwand der Views, die Qualität des Codes gelitten hat. Ich muss jetzt jedoch nach vorne schauen und versuchen, die restlichen Arbeiten qualitätsgerecht zu erledigen.

## Dienstag 27. November 2018 (3h)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag 1 (27.11.2018) | | | | |
| Phase: Realisieren & Kontrollieren | | | | |
| Tätigkeit | **Soll-Zeit (Stunden)** | **Ist-Zeit (Stunden)** | **Bemerkung** |
| Realisation dokumentieren | 2 | 2 |  |
| Code Quality steigern | 2 | 2 | Musste zur eigenen Zufriedenheit noch reingequetscht werden. |
| Test durchführen | 2 | 2 |  |
| Testprotokoll & Fazit erstellen | 1.5 | 1.5 |  |
| Arbeitsjournal schreiben | 1 | 1 |  |
| Total | 8.5 Stunden | 8.5 Stunden |  |

Tabelle 7: Arbeitsprotokoll 27.11.2018

Da ich am Freitag eigentlich die ganze Implementation der Arbeit gemacht habe, ging es heute nun daran diese zu dokumentieren. Nachdem ich meine Ergebnisse dokumentiert hatte ging es weiter mit der Phase «Kontrollieren». Hierbei habe ich meine Abnahmetests durchgeführt und ein Testprotokoll erstellt.

### Erfolge

Obwohl die Zeit für die Realisationsphase abgeschlossen ist, hatte ich heute noch 2 Stunden Zeit, um gewisse Verbesserungen zu machen. Hauptsächlich habe ich an der Aufteilung der Klassen gearbeitet. Nun sieht es so aus, dass die Code Qualität in einem Zustand ist, der gerade noch akzeptable ist. Hätte ich noch einen Tag mehr Zeit, könnte ich alle Code Smells beheben. Sehr zufrieden bin ich mit der Funktionalität der Applikation. Ich konnte alle Tests erfolgreich durchführen und auch das Look & Feel der Applikation ist akzeptable.

### Fazit

Ich habe für jeden Tag, während dem Projekt, eine Stunde am Schluss freigelassen. Heute habe ich richtig gemerkt, wie es mir den Tag gerettet hat. Ich kam nie wirklich in Stress, da ich immer diese freie Stunde vor Augen hatte. Nicht wie in der letzten Probe-IPA, wo ich mit dem Zeitdruck gar nicht mehr arbeiten konnte (Arbeitsleistung fällt bei mir mit Zeitdruck). Ausserdem konnte ich diese freie Zeit einsetzen, um den Mehraufwand während des Erstellens der Benutzeroberfläche, zu kompensieren.

## Mittwoch 28. November 2018 (3h)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tag 1 (28.11.2018) | | | | |
| Phase: Auswerten | | | | |
| Tätigkeit | **Soll-Zeit (Stunden)** | **Ist-Zeit (Stunden)** | **Bemerkung** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Total |  |  |  |

Tabelle 8: Arbeitsprotokoll 28.11.2018

### Erfolge

### Misserfolge

### Massnahmen

### Fazit

# Kurzfassung

In der Kurzfassung wird kurz und bündig das Kapitel «Teil 2: Projekt» zusammengefasst. Es zeigt die wichtigsten Stichpunkte aus diesem Teil und gibt dem Leser einen raschen Überblick der Arbeit.

## Ausgangslage

## Umsetzung

## Ergebnis

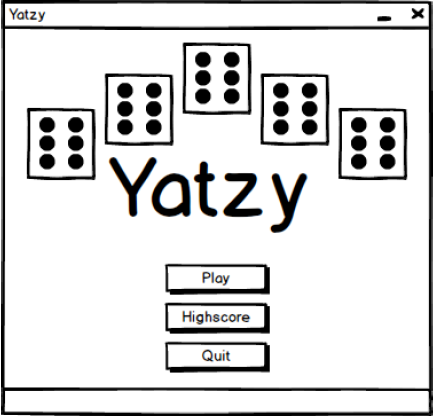
# Informieren

Das Informieren ist die erste Stufe in der gewählten Projektmethode «IPERKA». In diesem Schritt geht es vor allem um die Auftragsanalyse. Hier werden benötigte Information herausgesucht und zusammengestellt. Dabei werden Kriterien wie Umgebung, Bedingungen, Vorkenntnisse, Ressourcen und Ziel ganz genau hinterfragt. Man versucht in diesem Schritt den Auftrag ganz genau zu untersuchen, damit man bereits offene Fragen abdecken und mögliches Fehlerpotenzial aufweisen kann, um einen groben Überblick zu erhalten. Dieser Schritt dient als Grundlage für das nächste Kapitel «Planung».

## Ist-Analyse

Als Übung für die richtige IPA, haben gewisse Lehrling, von Torben Dziuk, den Auftrag erhalten, eine Probe-IPA zu schreiben. Als Aufgabe wurde eine reale Aufgabe der ICT-Skills Schweiz gewählt. Explizit stammt diese Aufgabe aus den Regionalmeisterschaften des Jahres 2017. Die Aufgabenstellung wurde als PDF übergeben und muss vom Kandidaten umgesetzt werden. Das Projekt wird von Grund auf aufgebaut und es existieren keine Vorarbeiten, für die Realisation der Aufgabenstellung. Da der Inhalt des Projektes ganz genau in der Aufgabenstellung (siehe Anhang) definiert wurde, bedarf es keiner weiteren Analyse für den Ist-Zustand.

## Soll-Analyse

Die Soll-Analyse ist eine genauere Analyse der Anforderungen und zeigt den Zustand, welcher als Ziel definiert wurde.

### Frontend

Soll-Analyse der Frontend-Implementation.

***Menu***

Wenn man die Applikation startet, sollte als erstes das Menu angezeigt werden. Dieses Fenster dient zur Navigation durch die Applikation. Von diesem Fenster aus kann man die verschiedenen Ansichten der Applikation öffnen. Ebenfalls, sobald einer dieser weiteren Ansichten geschlossen wird, soll man auf das Menu zurückgeleitet werden.

***Game***

Abbildung 2: Menu

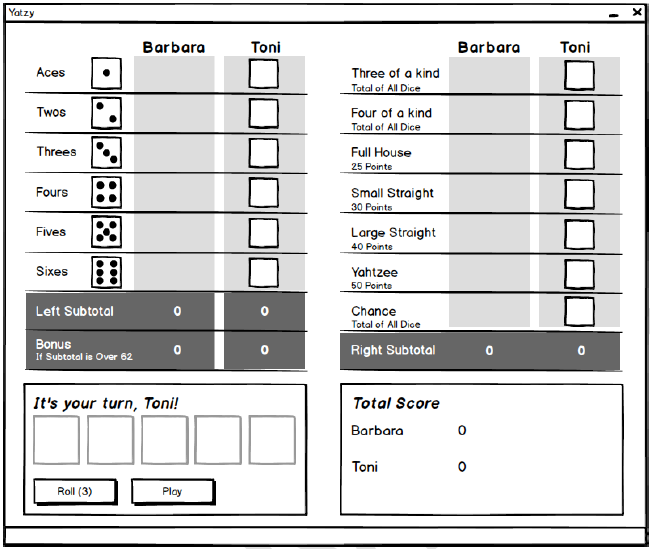
Wenn vom Menu aus auf den Button «Play»   
gedrückt wird, soll sich die Spielerauswahl öffnen. Hier soll man zwei Spieler eintragen können, welche im nächsten Schritt ein Yatzy-Duell ausführen können.

Abbildung 3: Yatzy-Spielbrett

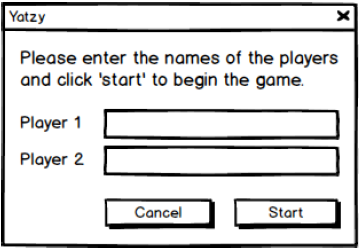
Nachdem man die Spieler eingetragen hat, und auf den Button «Start» geklickt hat, soll man auf das «Spielbrett» weitergeleitet werden. Das Spielbrett ist ein Yatzy-Spielbrett für zwei Spieler. In der Aufgabenstellung im Anhang wird genauer beschrieben, wie Yatzy funktioniert.

Abbildung 4: Spielerauswahl

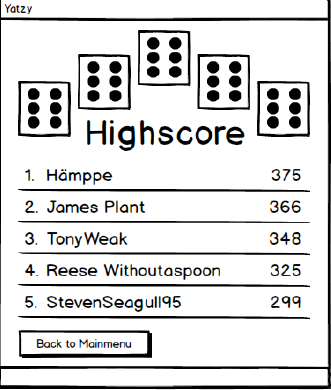
***Highscore***

Abbildung 5: Highscore

Diese Ansicht soll sich öffnen, wenn man auf dem Menu auf den Button «Highscore» klickt. Auf der Highscore-Ansicht sieht man jeweils die fünf Spieler mit den höchsten erreichten Punkten. Wenn man auf diese Ansicht geht, soll eine Verbindung zur Datenbank hergestellt werden und die entsprechenden Daten müssen eingetragen werden.

### Backend

Alle Frontend-Schnittstellen interagieren mit der Backend-Implementation. Das Backend ist dafür zuständig die Logik der Applikation bereitzustellen und bei Bedarf die Daten in der Datenbank zu hinterlegen.  
In der Aufgabenstellung sind die Yatzy-Spielregeln genau beschrieben und müssen so implementiert werden. In der Datenbank soll es eine Tabelle mit den Highscores geben, welche eingelesen und manipuliert werden können.

## Systemübersicht

In diesem Projekt gibt es eigentlich nur zwei Systeme welche miteinander kommunizieren. Es gibt die Client-Applikation welche jeweils auf die MSSQL-Datenbank zugreift. Dazwischen sitzt das Entity Framework, welches als ORM (Object-relational Mapper) fungiert und die Client-Applikation bei Datenbankabfragen unterstützt.

*Entity Framework*

**.NET Framework**

**(Client Application)**

**WPF**

**MSSQL**

**C#**

**SQL-Datenbank**

Abbildung 6: Systemübersicht

Wie man in der obigen Abbildung sehen kann, wird innerhalb des .NET-Framework mit C# und WPF gearbeitet. WPF wird für das Frontend, respektive für die Benutzeroberfläche, genutzt und C# für die Business-Logik und Daten-Aufbereitung der Benutzeroberfläche.

# Planen

Im zweiten Schritt der IPERKA-Methode werden nun die möglichen Lösungsvarianten und das Vorgehen ausgearbeitet. Es werden einzelne Arbeitsschritte geplant, um einen gesamten Arbeitsablauf zu erstellen. Im Arbeitsablauf werden Hilfsmittel und Werkzeuge ermittelt, sowie der Zeitbedarf für die Arbeitsschritte geschätzt. Zudem werden Qualitäts-Kriterien für den Schritt «Kontrollieren» gesammelt.

## Use Cases

In diesem Kapitel beschreibe ich die verschiedenen Anwendungsfälle anhand von Use Cases. Bei Use Cases ist es nicht nur wichtig, dass man darstellt, welche Funktionalität von wem aufgerufen werden kann, sondern auch was nach dieser Aktion passiert und ob es Vorbedingungen gibt.

In meiner Applikation gibt es eigentlich nur eine Rolle – Den Spieler. Dieser interagiert mit der ganzen Applikation. Er hat keine Einschränkungen und ist vollkommen dazu berechtigt jede Funktionalität aufzurufen. Da es für ein Yatzy-Spiel jedoch mindestens zwei Leute braucht, erscheint eine neue Rolle «Spieler 2» sobald man das Spiel startet. Diese Rolle ist nur verfügbar, während das Yatzy auf dem Spielbrett gespielt wird. Sobald das Spiel vorbei ist, gibt es wieder nur eine Rolle

Abbildung 7: Use Case-Diagram

|  |  |
| --- | --- |
| Use Case 1 – Highscore aufrufen | |
| Beschreibung | Als Spieler möchte ich auf die Highscores von anderen Spieler zugreifen können. |
| Konsequenz | Man wird auf die Ansicht «Highscore» weitergeleitet und kann die besten Punktzahlen der anderen Spieler einsehen. |
| Ablauf | 1. Auf dem Menu sein. 2. Knopf «Highscores» klicken. 3. Seite «Highscores» wird angezeigt. |

Tabelle 9: Use Case Highscore

|  |  |
| --- | --- |
| Use Case 2 – Applikation beenden | |
| Beschreibung | Als Spieler möchte ich die Applikation beenden können |
| Konsequenz | Die Applikation wird geschlossen. |
| Ablauf | 1. Auf dem Menu sein. 2. Knopf «Quit» klicken. 3. Applikation wird geschlossen. |

Tabelle 10: Use Case Applikation beenden

|  |  |
| --- | --- |
| Use Case 3 – Spieler auswählen | |
| Beschreibung | Als Benutzer möchte ich zwei Spieler für das Yatzy-Spiel erfassen können. |
| Konsequenz | Spielererfassung wird geöffnet und Spieler können bereitgestellt werden. |
| Ablauf | 1. Auf dem Menu sein. 2. «Play» klicken. 3. Spielererfassung wird geöffnet 4. Spieler erfassen |

Tabelle 11: Use Case Spieler auswählen

|  |  |
| --- | --- |
| Use Case 4 – Yatzy Spielen | |
| Beschreibung | Als Spieler möchte ich das Spiel «Yatzy» auf einem angemessenen Spielbrett spielen können. |
| Vorbedingung | Es müssen zwei Spieler für das Yatzy bereitgestellt werden. |
| Konsequenz | Yatzy-Spielbrett wird geöffnet und es kann gespielt werden. |
| Ablauf | 1. Auf dem Menu sein. 2. «Play» klicken. 3. Spieler für Yatzy erfassen. 4. Spielbrett wird geöffnet. 5. Yatzy spielen. |

Tabelle 12: Use Case Yatzy spielen

## Testkonzept

Das Testkonzept wird in der Planung erstellt und wird danach in der Phase «Kontrollieren» wiedergenutzt. Im Testkonzept werden alle gewünschten Funktionalitäten festgehalten, damit später sichergestellt werden, dass die Vorgaben eingehalten wurden. Um dieses Prozedere zu ermöglichen werden verschiedene Testfälle erstellt.

### Abnahmetests

Für die Sicherung der Qualität und der Anforderung, werden verschiedene Abnahmetest definiert. Unter einem Abnahmetest versteht man in der Softwaretechnik die Testfälle, welche gebraucht werden um sicherzustellen, dass die Muss-Kriterien einer Applikation erfüllt sind.

**Testfälle**

Anbei werden verschiedene manuelle Testfälle erstellt, welche mit den Anforderungen in den Use Cases korrelieren. Die Testfälle werden nach den Use Cases benannt (Nummer + Bezeichnung).

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 1) Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 1.1 | Applikation wurde gestartet | 1. «Highscore» wird gedrückt. 2. Highscore-Seite wird angezeigt | Die 5 besten Punkzahlen der erfassten Spieler werden der Reihe nach angezeigt. |
| 1.2 | Seite «Highscore» ist geöffnet | 1. Auf «Back to Mainmenu» klicken | Spieler wird auf Menu zurückgeleitet. |

Tabelle 13: Testfälle Use Case 1

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 2) Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 2.1 | Applikation wurde gestartet | 1. Auf «Quit» klicken | Die Applikation soll beendet werden |

Tabelle 14: Testfälle Use Case 2

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 3) Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 3.1 | Applikation wurde gestartet | 1. Auf «Play» klicken | Benutzerfassung wird angezeigt |
| 3.2 | Benutzererfassung ist geöffnet | 1. Spieler 1 eintragen 2. Spieler 2 eintragen 3. «Start» klicken | Spieler werden angenommen und Spielbrett wird geöffnet. |
| 3.3 | Siehe oben | 1. Spieler leer lassen 2. «Start» klicken | Eine entsprechende Fehlermeldung wird angezeigt. |
| 3.4 | Siehe oben | 1. «Cancel» klicken | Benutzerfassung wird geschlossen und Menu wird angezeigt |

Tabelle 15: Testfälle Use Case 3

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 4) Nr. | Vorbedingung | Testschritte | Erwartetes Ergebnis |
| 4.1 | Spieler sind erfasst | 1. Yatzy-Spielbrett öffnen | Ein nicht ausgefülltes Spielbrett wird angezeigt. Spieler soll notifiziert sein, dass er am Zug ist. (zufällig ausgewählter Spieler) Die Würfel sollen noch nicht angezeigt werden, da noch niemand gewürfelt hat.  Spieler hat für seinen Zug 3 Würfelversuche |
| 4.2 | Spielbrett ist geöffnet | 1. Spieler 1 klickt auf «Roll 3» | Würfel-Ergebnis wird angezeigt. |
| 4.3 | Spieler 1 hat 1. Mal gewürfelt | 1. Einen beliebigen Würfel auswählen und nochmals Würfeln | Nur noch 4 Würfel werden gewürfelt und der Spieler hat nur noch einen Versuch |
| 4.4 | Spieler hat bereits 3 Mal gewürfelt | 1. Spieler versucht nochmals zu würfeln | Der «Roll»-Button ist deaktiviert und es kann nur noch der Zug beendet werden. |
| 4.5 | 1.2 & 1.3 | 1. Beliebige Würfel auf eine Yatzy-Kombination setzen. | Spieler darf nur in seiner Spalte etwas setzen. Temporäres Ergebnis wird entsprechend angezeigt. |
| 4.6 | 1.4 | 1. Spieler klickt mit gewünschten Yatzy-Kombination auf «Play» | Zug wird beendet und Spieler 2 ist an der Reihe. Punktzahlen von Spieler 1 werden übernommen  Linker und Rechten Subtotal werden berechnet & Gesamtpunktzahl wird berechnet. |
| 4.7 | Yatzy Spielbrett geöffnet | 1. Spieler soll solange spielen, bis in der Linken spalte ein Subtotal von über 62 erreicht wird. | Der Spieler erhält einen Bonus von 35 Punkten. |
| 4.8 | Yatzy Spielbrett ist geöffnet | 1. Würfel-Kombination auf «Yatzy» setzen (Soll nicht Yatzy sein) | 0 Punkte werden für dieses Feld berechnet. |
| 4.9 | Alle Yatzy-Kombinationen sollen ausgefüllt sein | 1. Spiel abschliessen | Spieler mit höherer Punktzahl ist der Sieger.  Es wird dem Sieger gratuliert und gefragt, ob nochmals gespielt werden soll (Gleiche Spieler). Wenn Ja geklickt wird, wird das Spiel neu gestartet und falls Nein geklickt wird, wird der Spieler auf das Menu zurückgeleitet. Punktestand der beiden Spieler soll in der Datenbank erfasst werden (Für Highscore). |

Tabelle 16: Testfälle Use Case 4

## Aktivitätsdiagram

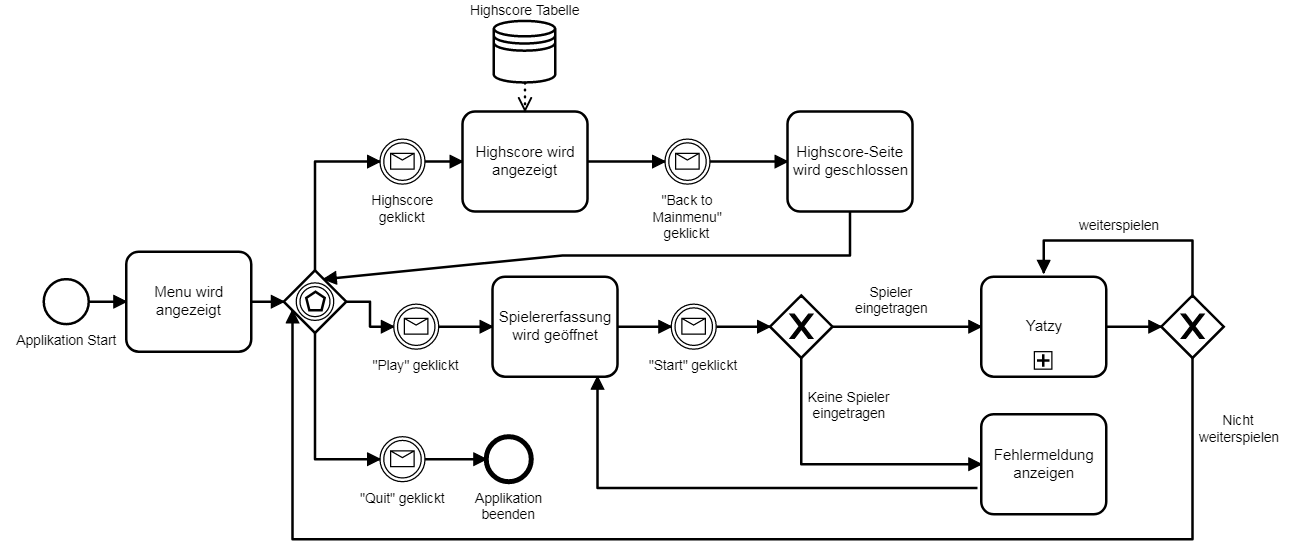
Neben den gewünschten Anforderungen an die Applikation, welche in den Use Cases abgebildet wurden, ist es auch wichtig, den gesamten Programmfluss zu planen. Dieser ist durch die Aufgabenstellung fast komplett definiert und wird in einem Aktivitätsdiagramm visualisiert, um dem Leser und dem Entwickler ein genaueres Verständnis des Ablaufs zu geben.

Abbildung 8: Aktivitätsdiagramm

***Sub-Routine «Yatzy»***

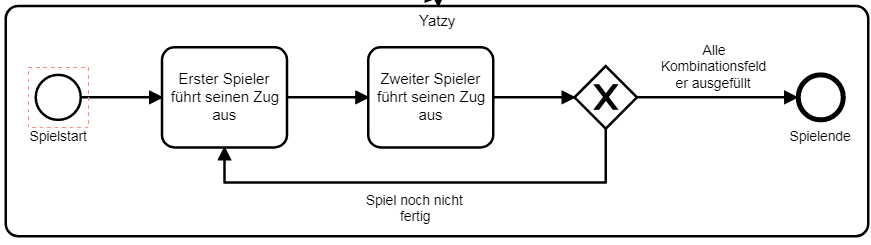


Abbildung 9: Aktivitätsdiagram Subroutine Yatzy

## Software-Architektur

In diesem Kapitel werden alle Themen bezüglich der allgemeinen Software-Architektur behandelt. Es werden die geplanten Komponenten definiert, ohne sie zu konkretisieren. Ich werde die funktionalen Verantwortungen der einzelnen Komponenten genauer beschreiben, jedoch ihre konkrete Implementation offenlassen, damit ich in der Phase «Realisieren» noch Anpassungsmöglichkeiten habe.

|  |  |  |
| --- | --- | --- |
| Komponente | Beschreibung | Abhängigkeiten |
| DatabaseService | Verknüpft die Models mit der Datenbank. Auf diesem Service werden alle Datenbankabfragen aufgerufen. | Entity Framework |
| Models | Korrespondieren mit Datenbanktabellen. Behälter für die Daten | Keine Spezialitäten |
| ViewModels | Verbindung zwischen Models und View. Datenaufbereitung für Benutzerschnittstelle | Keine Spezialitäten (MVVM Framwork muss gewählt werden) |
| Views | Konkrete XAML Implementation mit Anknüpfung an ViewModels (DataBinding) | Keine Spezialitäten |
| DI-Service | Wrapper für IOC-Container. Zuständig für Dependency Injection. | Keine Spezialitäten (DI-Framework muss gewählt werden) |

Abbildung 10: Software-Komponenten

### Klassendiagramm

Für das allgemeine Verständnis des Entwicklers, wird eine grobe Planung der Klassenarchitektur benötigt. In einem Klassendiagramm werden die benötigten Klassen abgebildet und schon mit den gröbsten Methoden und Eigenschaften bestückt. Während der Realisierungsphase kann man immer wieder auf das Klassendiagramm zurückgreifen und hat somit einen Leitfaden während der Implementation. Im nachfolgenden Klassendiagramm werden nur die nötigsten Methoden definiert, damit während der Realisierungsphase Erweiterungen durchgeführt werden können.

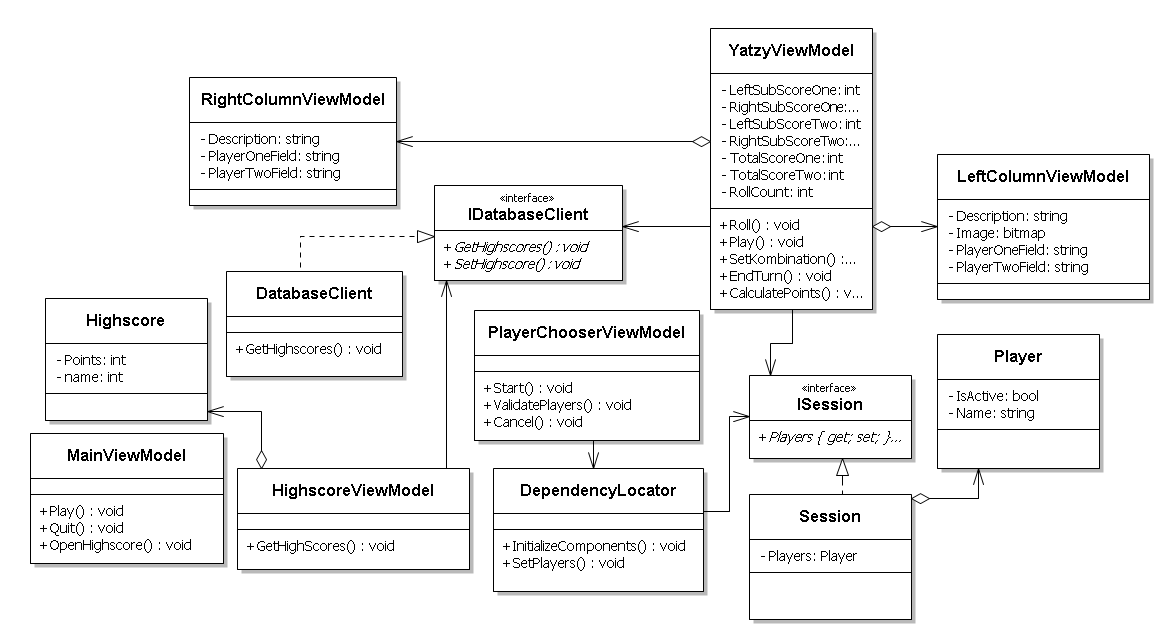


Abbildung 11: Klassendiagramm

## Datenbank-Architektur

Abbildung 12: "ERD"

In diesem Kapitel wird die Struktur für die Datenbank-Implementation geplant. Laut der Aufgabenstellung, benötigt die Applikation nur eine kleine Datenbank mit einer einzigen Tabelle. Die Highscores sollen in der Datenbank abgespeichert werden.

# Entscheiden

Damit man mit dem dritten Schritt von IPERKA beginnen kann, müssen zuerst die Anforderungen aus dem zweiten Schritt «Planen» erfüllt sein. Nach der Planung müssen die verschiedenen Lösungsvarianten gegenübergestellt werden und eine davon ausgewählt werden. Hierbei werden verschiedene Faktoren miteinander abgeglichen und ausgewertet.

## Architekturkonzept

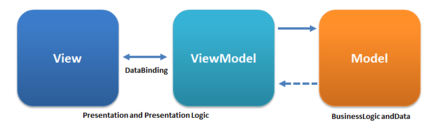
Ich arbeite seit 2 Jahren mit C# und entwickle seit je her mit dem Entwurfsmuster «MVVM» (Model View ViewModel). Anders als bei MVC (Model View Controller) trennt man die Darstellung von der Logik der Benutzerschnittstelle. Dieses Konzept zielt, unter anderem, auf WPF (Windows Presentation Foundation), da ein Datenbindungsmechanismus erforderlich ist. Wegen meiner Erfahrung in diesem Bereich und der Kompatibilität zu WPF, arbeite ich nach MVVM in meiner Applikation. (Wikipedia, 2018)

Abbildung 13: MVVM

## Benötigte Abhängigkeiten

Wenn man sich in der Objekt-Orientierten Programmierwelt befindet, stellt sich immer wieder die Frage, wie man die Abhängigkeiten eines Moduls, einer Klasse oder eines Service bereitstellt. Es gibt zwei Möglichkeiten die für mich in Frage kommen:

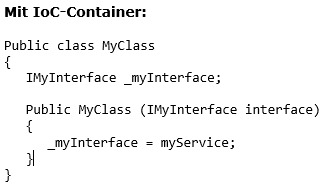
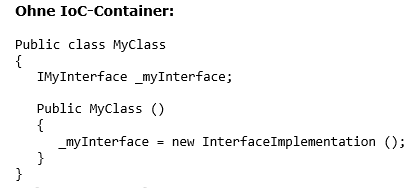
* Ich könnte normal wie üblich die Abhängigkeiten einer Klasse einfach mitübergeben. Somit müsste ich mich nicht wirklich um ein bestimmtes Konzept kümmern und kann mich auf die Logik konzentrieren. Hier laufe ich aber auf folgende Probleme: Tiefe Kohäsion, schlechte Testbarkeit, aufwändige Anpassungen.
* Als zweite Möglichkeit, könnte ich mit einem sogenannten IoC-Container arbeiten, welcher für mich die Abhängigkeiten auflöst und bereitstellt (Dependency Injection). Hierbei muss ich jedoch ein bestimmtes Konzept befolgen und bin eventuell auf weitere externe Abhängigkeiten angewiesen.

Abbildung 14: Mit IoC Beispiel

Abbildung 15: Ohne Ioc Beispiel

Eine konkrete Implementation des Interfaces wird vom IoC aufgelöst und übergeben.

Ich habe mich schlussendlich für die Version mit einem IoC-Container entschieden, da ich schon einige Zeit mit der IoC-Variante von Unity arbeite und mich relativ gut mit dem Konzept auskenne. Ausserdem lege ich bei dieser Arbeit hohen Wert auf die Qualität des Codes und ohne Dependency Injection ist dies sicherlich schwierig.

## Datenbank-Implementation

Bei Entity Framework gibt es verschiedenen Möglichkeiten, wie man die Datenbank, respektive die Datenbankverbindung, realisiert. Die zwei meist gebrauchten Methoden sind Code-First und Database-First. Wie eigentlich schon in den beiden Namen ersichtlich, wird bei der ersten Methode aus dem Code eine Datenbank generiert und bei der zweiten wird zuerst die Datenbank erstellt und danach der Code.

Bei Code-First ist der Vorteil, dass man mehr Kontrolle über die Struktur der Datenbank hat und bei Database-First ist das Problem, dass man zum Teil Schwierigkeiten hat, die Datenbank-Struktur anzupassen und den Code neu daraus zu generieren. Database-First wird oft verwendet, wenn man mit einem Legacy-System arbeitet, da man dort meistens gewisse Sicherheitsschritte durchführen muss, was bei Code-First die Migrationen unterbinden würde. Da ich nicht mit einem Legacy-System arbeite und mir die Code-First Variante besser liegt, arbeite ich mit der Code-First-Methode.

## Tools und Frameworks

Die meisten Tools und Frameworks, welche ich in diesem Projekt brauchen werde, sind schon in der Aufgabenstellung definiert und wurden während des Erstellens der Aufgabe definiert. Die untenstehenden Frameworks wurden während der Entscheidungsphase festgelegt.

### MVVM Light

MVVM ist wie oben beschrieben eine Software-Architektur. Es existieren, zur Hilfe der Umsetzung, verschiedene MVVM-Frameworks, welche einem unterstützen, die Architektur optimal umzusetzen. Aus meiner Abteilung kenne ich eigentlich zwei Frameworks: Prism & MVVM Light. Auf Grund der nachfolgenden Entscheidungsdarstellung, wähle ich für die Umsetzung MVVM Light.

|  |  |  |
| --- | --- | --- |
| Kriterium | Prism | MVVM Light |
| Mediator- Pattern Umsetzung | Benutzt IEventAggregator | Benutzt IMessenger: Bietet mehr Funktionalität gegenüber IEventAggregator (Z.B Messages mit Tokens versenden). |
| Testbarkeit | Prism ist ein wenig aufwendig zum Testen | MVVM-Light kann viel einfacher getestet werden. (Z.B. IMessenger kann einfacher gemockt werden). |
| Zweck | Prism ist eigentlich kein MVVM-Framework. Es ist ein allgemeines Architektur-Framework und bekam mit der Zeit den Namen eines MVVM-Frameworks, da es eine Art Richtlinie für die Umsetzung von MVVM enthält. Mit Prism kann man ebenfalls auch andere Architekturen umsetzen. | Voll und ganz MVVM-Framework |
| Community | Viele Foren und Beiträge vorhanden. | Beiträge im Internet sind ein bisschen mager, aber genügend vorhanden. |
| Einfachheit | Aufsetzen bisschen komplizierter. | Ganz simple. |

Tabelle 17: MVVM-Framework Entscheidung

### Unity

Ich habe mich bereits für die Verwendung eines IoC-Container entschieden. Es gibt diverse Container, welche das .NET-Framework unterstützen und für die Dependency Injection optimal sind. Da ich jedoch nur mit dem Framework «Unity» Erfahrung habe und das Evaluieren der verschiedenen Produkte ein Mehraufwand wäre, für den ich keine Zeit habe, entscheide ich mich für Unity.

# Realisieren

Die Realisierung und somit der vierte Schritt der IPERKA-Methode, ist die zeitintensivste Phase in den meisten Projekten. Hier werden die zuvor erfüllten Schritte von IPERKA (Planen & Entscheiden) aus der Theorie in die Praxis umgewandelt. Dabei ist es wichtig den Arbeitsablauf einzuhalten und nicht ohne zwingende Gründe zu ändern. In diesem Schritt ist es äusserst wichtig genügend Zeit für eine gute Reflexion einzuplanen, sodass man schnell auf Misserfolge reagieren kann und die Massnahmen daraus planen und umsetzen kann.

## Projekt aufsetzen

Zu Beginn musste ich erstmal das Projekt erstellen und mit GitHub verknüpfen. Dazu habe mit GitTortoise das Repository erstellt und als Erstes eine GitIgnore-Datei erstellt. In dieser Datei gibt man an, welche Dateien wirklich in das Repository eingecheckt (Hochladen in Versionierung) werden müssen und welche nicht. So hat man schlussendlich keine unnötigen Dateien, wie temporäre Dateien, Log-Dateien oder NuGet-Packages in seinem Repository.

Im nächsten Schritt habe ich alle benötigten NuGet-Packages dem Projekt hinzugefügt und eine allgemeine Verzeichnisstruktur erstellt.

Abbildung 16: Verzeichnisstruktur

## Aufsetzen der Datenbank

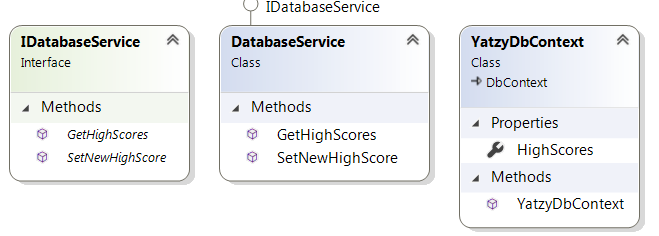
Es existiert im Endeffekt nur eine Tabelle in der Datenbank, doch ich wollte die Datenbank als Erstes aufsetzen, da ich schlechte Erfahrungen gesammelt habe, wenn man das zu einem späteren Zeitpunkt erledigt (Probleme mit Entity Framework).  
Da ich mit der Code-First-Methode arbeite, musste ich als erstes die Klassenabbildung der Tabelle erstellen (HighScore). Diese wurde nach dem bereits erstellten ERD erledigt. Als nächstes musste der DbContext erstellt werden. Der DbContext (In meinem Fall «YatzyDbContext») wird gebraucht um Abfragen an die Datenbank zu machen und enthält die verschiedenen DbSets (Ein DbSet repräsentiert eine Tabelle). Zum Schluss musste ich noch die Schnittstelle implementieren, welche genutzt wird um den DbContext anzusteuern und Abfragen weiterzuleiten. Folgendes kam dabei raus:

Abbildung 17: Datenbank-Implementation

## Aufbau der Architektur

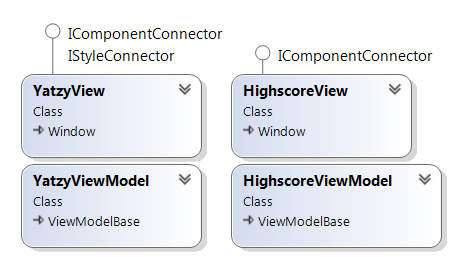
Ich musste mir bereits in der Phase «Planen» Gedanken machen, wie ich meine Applikation und vor allem die Struktur implementieren möchte. Dazu habe ich ein Klassendiagramm erstellt, welches mir während der Realisierung die Denkarbeiten abnimmt.  
Am Anfang war mir am wichtigsten, dass die allgemeine Struktur von MVVM umgesetzt wird. Dabei bin ich eigentlich immer gleich vorgegangen. Ich habe als erstes die View erstellt und habe dann gerade gesehen, welche Eigenschaften oder Commands das zugehörige ViewModel braucht und konnte es ganz einfach nachziehen. Im ganze Projekt sieht man den typischen Aufbau von MVVM. Jede View oder jedes User Control bekommt ein ViewModel zugeteilt auf welchem es arbeitet (Wird folgend noch genauer erklärt).  
Ein wichtiger Vorteil von MVVM ist sicherlich das Data Binding. Anstatt, dass ich jedes mal meinem UI-Element (Kontrollelement auf Benutzeroberfläche) einen neuen Wert übergebe, den es dann anzeigt, verbinde ich einfach das Element auf der Benutzeroberfläche mit einer Eigenschaft auf dem ViewModel. Sobald sich die Eigenschaft auf dem ViewModel ändert, wird die View notifiziert und erhält den neuen Wert. Das Beste daran ist, dass ich keine Koppelung zwischen View und ViewModel habe und somit jederzeit das eine auswechseln kann, ohne etwas im anderen zu ändern.

Abbildung 18:Beispiel View, ViewModel

**Dependency Locator**

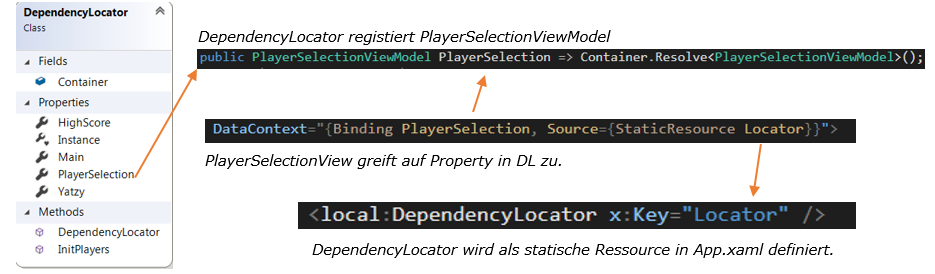
Das Data Binding alleine bringt uns momentan noch nicht sehr viel, da eine Benutzeroberfläche einen Datenkontext (das entspechende ViewModel) braucht, damit sie die verschiedenen Eigenschaften an sich «binden» kann. Es gibt verschiedene, schöne und nicht schöne, Varianten, wie man dies erzielen kann. Ich habe mich in der Phase «Entscheiden» für einen Unity-Container entschieden, welcher mir die Möglichkeit gibt, das ViewModel zu registieren und für den wiedergebrauch bereitzustellen. In meiner «Dependency Locator» - Klasse registriere ich zu Beginn alle benötigten ViewModels. Die Views müssen dann nur noch auf den Dependency Locator zugreifen, welcher als Statische Ressource bereitgestellt wird, und den Datenkontext auf das entsprechende ViewModel setzen.

Abbildung 19: Data Binding mit Dependency Locator

**Business-Logik (BL)**

Nachdem die grundlegende Applikationsstruktur erstellt wurde, fing ich an mit der Implementation der verschiedenen Funktionalitäten. Man kann natürlich die ganze BL in die ViewModels packen und dann würde es auch funktionieren, jedoch würde man so das erste Prinzip der SOLID-Prinzipien verletzen => Single Responsibility Principle. Dieses Prinzip besagt, dass eine Klasse jeweils nur eine Aufgabe haben soll. Somit wird das Verständnis erleichtert und es ist übersichtlicher. Also habe ich für jede Aufgabe, die es beim Yatzy gibt, eine eigene Klasse erstellt. Nachdem diese Klassen erstellt wurde, musst ich nur noch die Logik der Aufgabe implementieren und konnte sie mit den ViewModels verknüpfen.

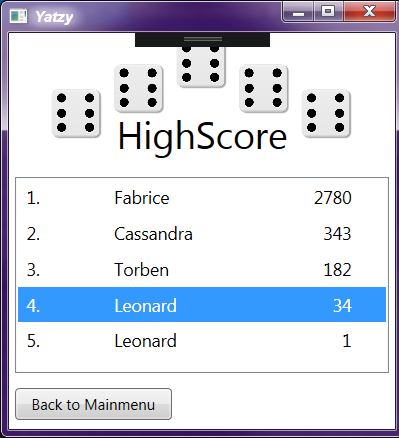
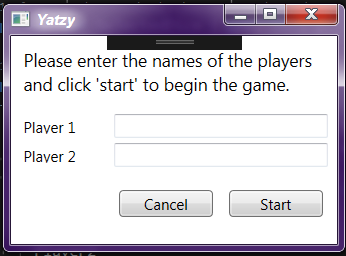
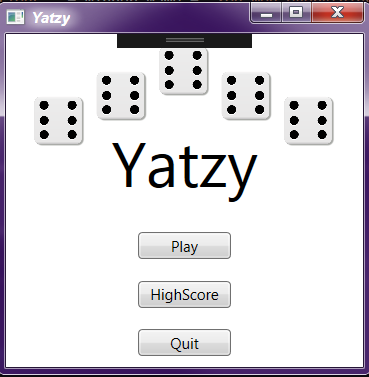
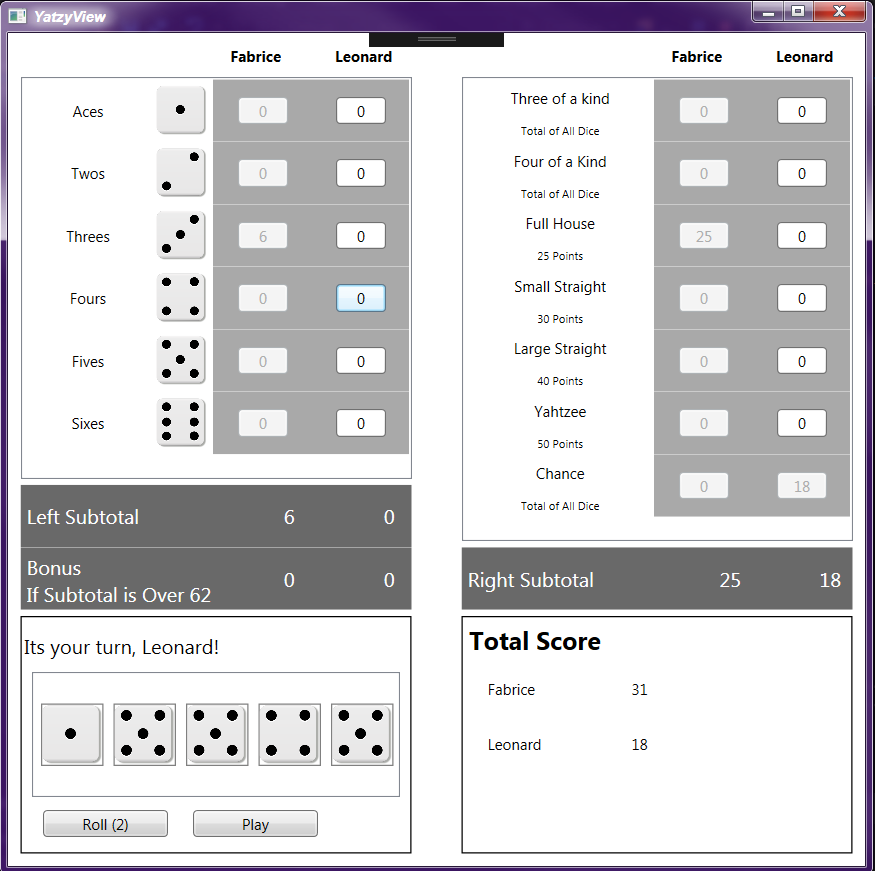
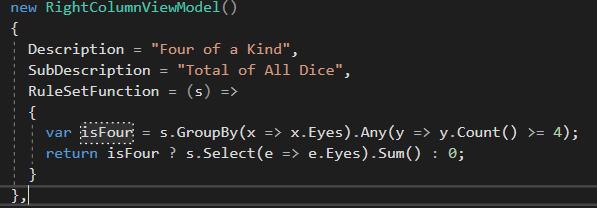
## Ergebnis und Erwähnenswertes

In diesem Kapitel wird das Ergebnis der Phase «Realisieren» dargestellt und es wird auf interessante Fakten während der Implementation verwiesen.

Das Projekt wurde, wie in der Aufgabenstellung beschrieben, umgesetzt und enthält jede funktionale Anforderung. Als Vorlage für die verschiedenen Benutzeroberflächen wurden die Wire-Frames in der Aufgabenstellung verwendet.  
Es existierten keine Styling-Vorgaben, jedoch wurde versucht, eine möglichst genaue Abbildung zu erstellen. Dies gelang nicht immer perfekt, doch die Benutzeroberflächen sind kompakt und einigermassen sauber umgesetzt worden. Es ist praktisch unmöglich, ein erstelltes Wire-Frame zu kopieren, da die Kontrollelemente je nach Tool anders sind.

**Benutzeroberflächen**

Die ersten drei Views sind ziemlich simple und eigentlich selbsterklärend.



Wie oben zu sehen ist, übergebe ich beim Erstellen der ViewModels eine Funktion (Callback) mit, welche die Punkte- / Validierungs-Logik definiert. Somit könnte ich jetzt noch weitere Punkteregeln für Yatzy definieren, ohne eine Änderung machen zu müssen.

Das Yatzy-Spielbrett hingegen ist ziemlich komplex und beinhaltet mehrere Verschachtelungen. Interessant hierbei ist der Aufbau der beiden markierten **Listen**. Ich habe für jedes Element in der Liste ein eigenes **ViewModel** definiert. Diese werden durch den BoardBuilder (Builder-Klasse für Erstellen des Spielbretts) zu Beginn des Spiels initialisiert.

Abbildung 20:Yatzy-Spielbrett

Abbildung 21: Menu

Abbildung 22: PlayerSelection

Abbildung 23: Highscores

# Kontrollieren

Der wichtigste Schritt in der IPERKA-Methode ist das Kontrollieren. Man kann seine Arbeit nicht abgeben, ohne dass man alles nochmals kontrolliert hat. Hierbei werden vor allem die Vorgaben, welche sich aus dem Schritt «Planen» ergeben haben, mit dem tatsächlichen Ergebnis verglichen und die Resultate festgehalten. Neben der Kontrolle des eigentlichen Produktes, ist es auch wichtig das Arbeitsverhalten zu überprüfen und allenfalls Abweichungen aus den Plänen aufzuzeigen.

## Meilensteine

Ich habe mir in der Zeitplanung gewisse Meilensteine definiert, welche ich als Leitfaden für die Realisierung genutzt habe. In diesem Schritt kontrolliere ich nun meine geplanten Meilensteine mit den wirklichen Projektdaten. Im Kapitel «Auswerten» werden die Soll- / Ist-Zeiten reflektiert.

|  |  |  |
| --- | --- | --- |
| Meilenstein | Soll-Datum | Ist-Datum |
| Projektbeginn | 21.11.2018 14:00 Uhr | 21.11.2018 14:00 Uhr |
| Informieren | 22.11.2018 10:00 Uhr | 32.11.2018 11:00 Uhr |
| Planen | 22.11.2018 15:00 Uhr | 22.11.2018 16:00 Uhr |
| Entscheidung | 22.11.2018 17:30 Uhr | 22.11.2018 17:30 Uhr |
| Realisieren | 27.11.2018 15:00 Uhr | 27.11.2018 15:00 Uhr |
| Kontrollieren | 27.11.2018 17:30 Uhr | 27.11.2018 17:30 Uhr |
| Auswerten | 28.11.2018 11:00 Uhr | 28.11.2018 11:00 Uhr |
| Projektabschluss | 28.11.2018 13:00 Uhr | 28.11.2018 13:00 Uhr |

## Testprotokoll

Im Kapitel «Planen» habe ich Testfälle zu den verschiedenen Use Cases erstellt. Beim Kontrollieren führe ich diese Testfälle aus und halte ihre Ergebnisse fest.

Ich verweise jeweils auf die Nummer der Testfälle im Kapitel «Testkonzept» (z. B. Testresultat 1.1 = Testfall 1.1) und stelle die Ergebnisse wie folgt dar.

**Bewertung**

Ich habe mir drei verschiedene Bewertung für einen Test überlegt:

* ✓ (Bestanden ohne Abweichung)
* - (Bestanden mit Abweichung)
* X (Nicht bestanden)

Wenn eine Abweichung bekannt ist, oder der Test fehlschlägt, dann muss im Testprotokoll die Abweichung beschrieben werden.

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 1) Nr. | Kurze Beschreibung | Resultat | Abweichung |
| 1.1 | Highscore anzeigen | ✓ | - |
| 1.2 | Auf Menu zurückgehen | ✓ | - |

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 2) Nr. | Kurze Beschreibung | Resultat | Abweichung |
| 2.1 | Applikation beenden | ✓ | - |

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 3) Nr. | Kurze Beschreibung | Resultat | Abweichung |
| 3.1 | Benutzererfassung öffnen | ✓ | - |
| 3.2 | Spielbrett mit validen Spieler öffnen | ✓ | - |
| 3.3 | Nicht erlaubte Spielernamen hinzufügen | ✓ |  |
| 3.4 | Benutzererfassung schliessen | ✓ |  |

|  |  |  |  |
| --- | --- | --- | --- |
| (Use Case 4) Nr. | Kurze Beschreibung | Resultat | Abweichung |
| 4.1 | Korrektes Spielbrett anzeigen | ✓ | - |
| 4.2 | Korrektes Würfeln | ✓ | - |
| 4.3 | Korrektes Würfeln mit einem Würfel selektiert | ✓ |  |
| 4.4 | Alle Würfel-Versuche aufgebraucht | ✓ |  |
| 4.5 | Ergebnis-Preview anzeigen | ✓ |  |
| 4.6 | Zug beenden | ✓ |  |
| 4.7 | Bonus | ✓ |  |
| 4.8 | Falsche Kombination gibt 0 Punkte | ✓ |  |
| 4.9 | Spielende | ✓ |  |

### Testfazit

# Auswerten

Im letzten Schritt von IPERKA reflektiert man nochmals, vom Informieren bis zum Kontrollieren die gesamte Bearbeitung des Auftrags. Beim Auswerten geht es um eine Selbsteinschätzung der erledigten Tätigkeiten. Man überlegt sich darüber hinaus was man aus der Arbeit gelernt hat, was man verbessern kann und wo man das Gelernte in der Zukunft anwenden kann.

## Reflexion

## Erkenntnisse und Erfahrungen

## Erweiterungen

## Fazit

# Glossar

Im Glossar werden die wichtigsten Fremdwörter und Technische Ausdrücke erklärt.

|  |  |
| --- | --- |
| **Begriff** | **Erklärung** |
| Vulnerability | Vulnerability, also eine Vulnerabilität, ist eine Schwachstelle in der Applikation. |
| Continious Integration (CI) | Kontinuierliche Integration beschreibt den Prozess, bei dem man fortlaufend Komponenten aus der Software zusammenfügt. CI wird vor allem für die stetige Steigerung der Softwarequalität genutzt. |
| Model View ViewModel (MVVM) | MVVM ist eine Beschreibung einer Architektur in der Software-Programmierung |
| Daily | Ein Daily ist eines der Scrum Artefakte. Ein Daily ist ein tägliches Meeting innerhalb des Entwicklerteams, wo über abgeschlossene und bevorstehende Arbeiten geredet wird. |
| ORM | Object-Relational Mapping, ist ein Konzept zum Abbildung von Objekten in einer relationalen Datenbank. |
| Wire-Frames | Wire-Frames werden dazu benutzt um eine konzeptionelle Abbildung einer Benutzeroberfläche darzustellen. Dabei wird wenig auf Design und Funktionalität geachtet, sondern man konzentriert sich auf die Anordnung der Elemente |
| UI-Controls | UI (User Interface) ist die Bezeichnung für die Benutzeroberfläche. UI-Controls sind die angewendeten Bausteine, für die Benutzeroberflächen-Programmierung. |
| Inversion of Control (IOC) | Dieses Programmierparadigma beschreibt die Arbeitsweise einer Einheit, eines Programms, oder eines Frameworks. In meinem Beispiel: Statt, dass die einzelnen Klassen sich um ihre Abhängigkeiten kümmern, wird die Kontrolle einer anderen Einheit übergeben. Sie regelt den Kontrollfluss und löst die Abhängigkeiten auf. |
| Command | Ein Command ist ein Befehl. Wenn ein Benutzer etwas klickt, führt er einen Befehl aus, welcher in der Applikation etwas auslöst. |
| Kohäsion | Kohäsion beschreibt, wie gut eine Einheit seine Logik abbildet. In der Programmierung versucht man eine möglichst hohe Kohäsion zu erreichen. Das heisst, dass jede Einheit(Klasse) nur für eine Aufgabe zuständig ist. |
| Mocks | Mocks, meistens Mockups, sind Vorführmodelle, welche in der Programmierung zur Modellierung von gewissen Einheiten genutzt werden. |
| Mediator- Pattern | In der Softwareentwicklung, definiert das Mediator-Pattern ein Objekt, welches vorgibt, wie eine Sammlung von Objekten untereinander interagiert. |
| Repository | Ein Repository ist ein verwaltetes Verzeichnis zur Speicherung und Kontrolle von Daten (Digitales Archiv). |
| SOLID | SOLID ist in der Objekt-Orientierten Programmierung eine Gruppe von Clean Code Prinzipien. Clean Code Prinzipien sind Richtlinien zur Verbesserung der Code Qualität, Testbarkeit, Lesbarkeit, Erweiterbarkeit etc. |

Tabelle 18: Glossar

# Verzeichnisse

Anbei werden alle verwendeten Ressourcen aufgelistet.

## Quellenverzeichnis

Developer, C. C. (2015 ). Clean Code Prinzipien. Abgerufen am 09. November 2018 von https://clean-code-developer.de/

mackenir. (2009). Databinding TabControl. Abgerufen am 07. November 2018 von https://stackoverflow.com/questions/686074/wpf-tabcontrol-databinding

S.A, S. (2008). Leading Product for CI. Abgerufen am 30. Oktober 2018 von https://www.sonarqube.org/

VSSM, V. s. (2014). Die 6-Schritte-Methode. Abgerufen am 24. Oktober 2018 von https://www.vssm.ch/sites/default/files/vssm/BB/docs/AusbildungEFZ/IPERKA\_1\_4\_D2017.pdf

Wikipedia. (16. September 2018). MVVM. Abgerufen am 01. November 2018 von https://de.wikipedia.org/wiki/Model\_View\_ViewModel

## Abbildungsverzeichnis

[Abbildung 1: Gantt-Diagramm 8](#_Toc531078180)

[Abbildung 2: Menu 15](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078181)

[Abbildung 3: Yatzy-Spielbrett 15](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078182)

[Abbildung 4: Spielerauswahl 15](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078183)

[Abbildung 5: Highscore 16](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078184)

[Abbildung 6: Systemübersicht 16](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078185)

[Abbildung 7: Use Case-Diagram 17](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078186)

[Abbildung 8: Aktivitätsdiagramm 21](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078187)

[Abbildung 9: Aktivitätsdiagram Subroutine Yatzy 21](#_Toc531078188)

[Abbildung 10: Software-Komponenten 22](#_Toc531078189)

[Abbildung 11: Klassendiagramm 22](#_Toc531078190)

[Abbildung 12: "ERD" 22](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078191)

[Abbildung 13: MVVM 23](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078192)

[Abbildung 15: Ohne Ioc Beispiel 23](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078193)

[Abbildung 14: Mit IoC Beispiel 23](file:///C:\Arbeitsordner\_ImportantStuff\Torben_Dziuk_stuff\2teProbeIPA\ProbeIPATorben\Doc\ProbeIPA_FabriceBossahard_28.11.2018.docx#_Toc531078194)

## Tabellenverzeichnis

[Tabelle 1: Änderungshistorie 1](#_Toc531078195)

[Tabelle 2: Beteiligte Personen 6](#_Toc531078196)

[Tabelle 3: Meilensteine 7](#_Toc531078197)

[Tabelle 4: Arbeitsprotokoll 21.11.2018 9](#_Toc531078198)

[Tabelle 5: Arbeitsprotokoll 22.11.2018 10](#_Toc531078199)

[Tabelle 6: Arbeitsprotokoll 23.11.2018 11](#_Toc531078200)

[Tabelle 7: Arbeitsprotokoll 27.11.2018 12](#_Toc531078201)

[Tabelle 8: Arbeitsprotokoll 28.11.2018 13](#_Toc531078202)

[Tabelle 9: Use Case Highscore 17](#_Toc531078203)

[Tabelle 10: Use Case Applikation beenden 17](#_Toc531078204)

[Tabelle 11: Use Case Spieler auswählen 18](#_Toc531078205)

[Tabelle 12: Use Case Yatzy spielen 18](#_Toc531078206)

[Tabelle 13: Testfälle Use Case 1 19](#_Toc531078207)

[Tabelle 14: Testfälle Use Case 2 19](#_Toc531078208)

[Tabelle 15: Testfälle Use Case 3 19](#_Toc531078209)

[Tabelle 16: Testfälle Use Case 4 20](#_Toc531078210)

[Tabelle 17: MVVM-Framework Entscheidung 24](#_Toc531078211)

[Tabelle 18: Glossar 28](#_Toc531078212)

# Aufgabenstellung

# Programmcode