[Datum]

Cedric Gasser | Vincent Dittl

Modul 226B / 120

BBZW Sursee

Hello World

Projektarbeit MVVM

Inhaltsverzeichnis

[1 Einführung 2](#_Toc42255700)

[2 Planung 2](#_Toc42255701)

[2.1 User Stories 2](#_Toc42255702)

[2.1.1 Anfang 2](#_Toc42255703)

[2.1.2 UI 2](#_Toc42255704)

[2.1.3 Shop 2](#_Toc42255705)

[2.1.4 Help 2](#_Toc42255706)

[2.1.5 Menu 2](#_Toc42255707)

[2.1.6 Start Menu 2](#_Toc42255708)

[2.2 Arbeitspaketeeinteilung 3](#_Toc42255709)

[3 GUI 3](#_Toc42255710)

[3.1 Mockup 4](#_Toc42255711)

[3.2 Bedienkonzept 6](#_Toc42255712)

[3.3 Prozessdiagramm 6](#_Toc42255713)

[4 Lernjournal 6](#_Toc42255714)

[4.1 22 Mai 2020 6](#_Toc42255715)

[4.2 27 Mai 2020 6](#_Toc42255716)

[4.3 28 Mai 2020 7](#_Toc42255717)

[4.4 22 Mai 2020 7](#_Toc42255718)

# Einführung

Wir wollten ein “Clicker Game” erstellen. Man kann klicken, um mehr von der In-Game Währung zu erhalten, mit dieser Währung kann man sich dann Maschinen kaufen die automatisch für einen klicken oder diese Währung produzieren.

Mit dem Grundprinzip ausgearbeitet, wollten wir ein Thema für unser Game festlegen:

* Man ist eine AI und möchte “Hello World” ausgeben.
* Für jedes ausgegebene “Hello World” bekommt man Karma, Karma ist die Währung.
* Am Anfang muss man print(“Hello World”) in einer Konsole eingeben.
* Mit Upgrades kann aus der Konsole ein GUI mit einem Button werden damit es schneller geht.
* Irgendwann kann man sich ein Programm kaufen, mit welchem automatisch “Hello World” ausgegeben wird.
* Sobald man genügend “Hello World”s pro Sekunde hat, wird das Heimnetzwerk freigeschaltet. (Im Shop sind nun andere Geräte verfügbar)
* Mit genügend Karma kann man andere Geräte kaufen, und dort “Hello World” ausgeben.

# Planung

## User Stories

### Hello World ausgeben

Priorität 1

|  |  |
| --- | --- |
| Als | Spieler |
| Kann ich | Auf einen Button klicken |
| um | Ein Hello World auszugeben und ein Karma zu bekommen |

#### Akzeptanzkriterien

UI

* Einen Button zum klicken welcher das es Karma erhöt und ein Hello World “printed”
* Ein Feld, welches jedes Hello World “printed”
* Die Anzahl Karma sollte angezeigt werden
* Die Anzahl Hello World per Second sollte angezeigt werden

### Shop

Priorität 2

|  |  |
| --- | --- |
| Als | Spieler |
| Kann ich | Im Shop Devices kaufen |
| um | Hello World’s automatisch zu generieren |

#### Akzeptanzkriterien

UI

* Alle Devices welche exisitieren werden aufgelistet
* Neben dem device Namen steht der Preis in Karma, die Hello World per Second und die Anzahl wie viel du besitzt
* Es hat einen Buy Button, welcher dir ein Device kauft und die Anzahl wie viel du besitzt erhöht

### Menu

Priorität 4

|  |  |
| --- | --- |
| Als | Spieler |
| Kann ich | Das Menu öffnen |
| um | Den Spielstand zu speichern oder zum Menu/Desktop zurückzukehren. |

#### Akzeptanzkriterien

UI

* Das Fenster ist im Vordergrund und kann nicht in den Hintergrund
* Es hat einen Button, welcher beim klicken den Spielstand in ein File Speichert und dich auf das Start Menu weiterleitet

Speichern

* Der Spielstand wird in einem Json File gespeichert.

### Start Menu

Priorität 5

|  |  |
| --- | --- |
| Als | Spieler |
| Kann ich | Das Start Menu öffnen / Das Spiel starten |
| um | Ein neues Spiel zu starten oder einen bestehenden Spielstand zu laden |

#### Akzeptanzkriterien

UI

* Es hat einen Button, welcher ein neues Spiel erstellt.
* Es hat einen Button, welcher ein bestehender Spielstand lädt
* Es hat einen Button, welcher das Programm beendet

Laden

* Der Spielstand wird von einem Json File geladen.

### Daten Laden

Priorität 3

|  |  |
| --- | --- |
| Als | Benuter |
| Kann ich | Json Files ändern |
| um | Beim Start vom Programm spezifische Objekte zu laden |

#### Akzeptanzkriterien

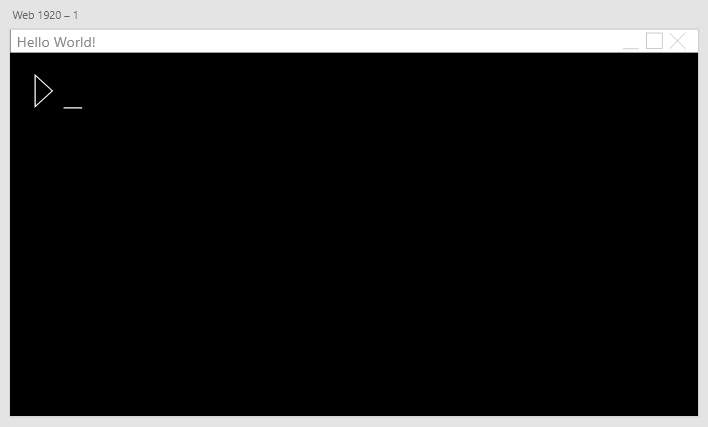
* Beim Starten vom Programm, werden die Objekte welche Hello Worlds produzieren von einem Jsonfile geladen.
* Jeder der das File bearbeiten darf, kann somit neue Objekte Hinzufügen oder alte löschen.

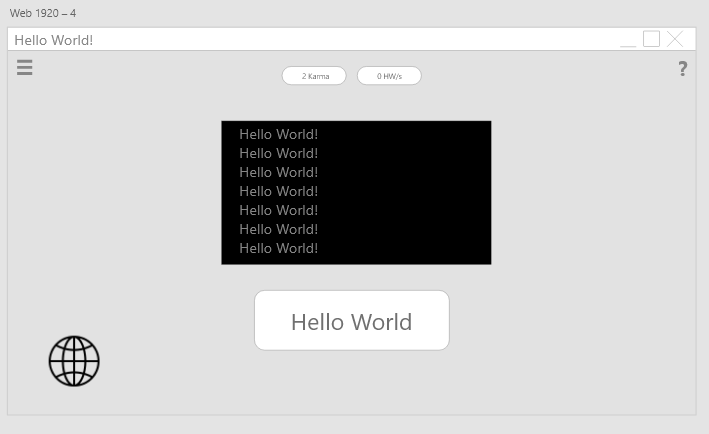
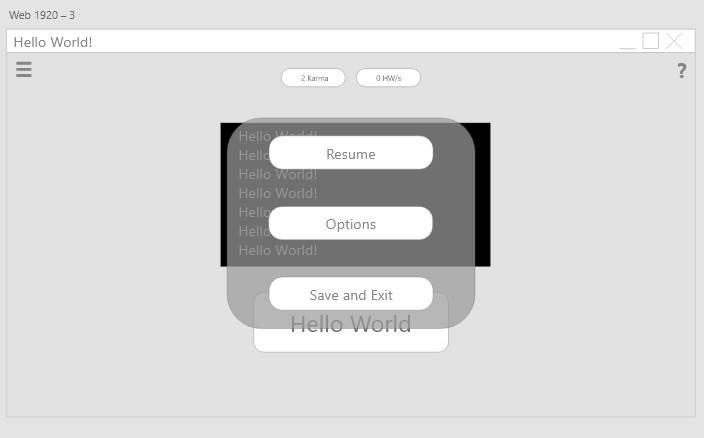
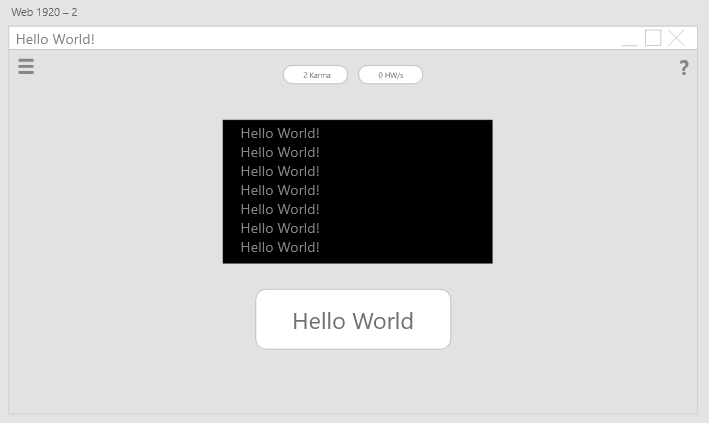
## Arbeitspaketeeinteilung

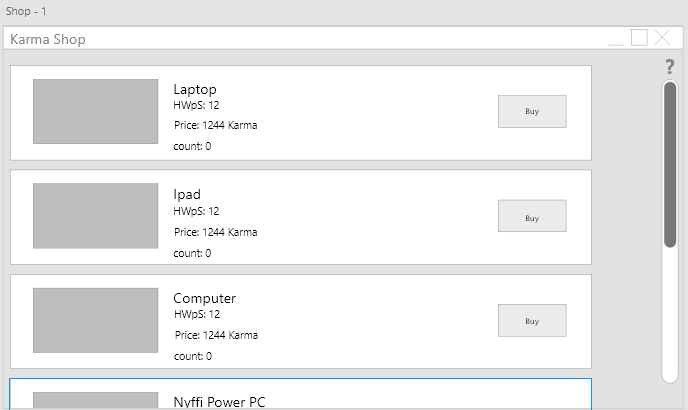
|  |  |
| --- | --- |
| Vincent | Cedric |
| GUI Mockups | Prozessdiagramm |
| User Stories | Bedienkonzept |
| Lernjournal | Lernjournal |
|  |  |

# GUI

## Mockup

KonsolenView:

GameView:



## Bedienkonzept

Es soll 2 Views geben:

### KonsolenView

Eine View die eine Konsole darstellt soll am Anfang eines Spiels gezeigt werden. In dieser View können Funktionen des Programmes, wie z.B. das Ausgeben von “Hello World” oder das Upgraden der Benutzeroberfläche mit Texteingaben wie **print “Hello World!”** oder **upgrade** aufgerufen werden.

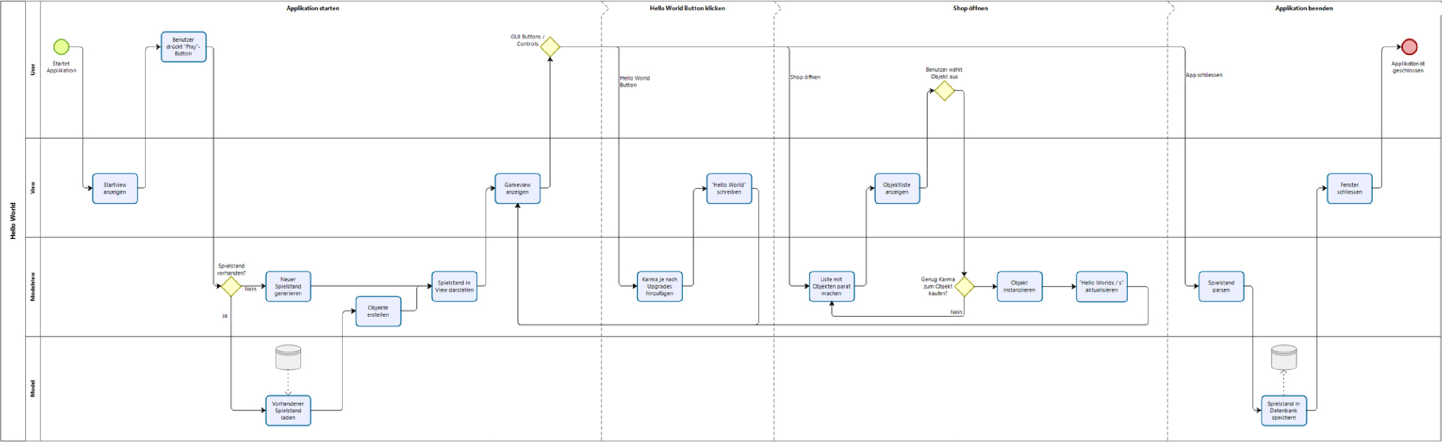
|  |  |
| --- | --- |
| Element | Zweck |
| Konsolenheader | Zeigt aktuelles Karma und die möglichen Befehle an |
| Eingabebereich | Können die Befehle, die im Header angegeben sind, eingegeben werden. |
| >print “Hello World!” | Befehl der ein Hello World ausgibt, damit bekommt man 1 Karma |
| >upgrade | Sobald genügend Karma vorhanden ist, kann mit diesem Befehl von dem KonsolenView zu dem eigentlichen GameView gewechselt werden |
| >exit | Das Spiel wird beendet, aber nicht gespeichert, denn bei einem Spielstand wo man noch immer im KonsolenView ist, lohne es sich nicht zu speichern. |

### GameView

Eine andere View ist dann die eigentliche GameView, mit den Informationen zum Karma und zu den Hello Worlds /s und Buttons für den Shop, das Menu und einem Button um “Hello World!” auszugeben. Eventuell soll in dieser View auch jedes “Hello World!” in einem separaten Ausgabeterminal ausgegeben werden.

|  |  |
| --- | --- |
| Element | Zweck |
| Hello World! - Button | Mit diesem Button wird ein “Hello World!” ausgegeben. |
| Ausgabeterminal | Hier werden alle “Hello World!” ausgegeben, zum Darstellen was gerade passiert. |
| HW/s - Anzeige | Zeigt an, wieviel “Hello World!” pro Sekunde von den gekauften Devices produziert werden |
| Karma - Anzeige | Zeigt an wieviel Karma man hat |
| Shop - Button | Mit diesem Button wird der Shop mit einer Liste aller Devices angezeigt, die man kaufen kann. |
| Menu - Button | Über diesen Button wird ein kleines Menu angezeigt, in welchem man den Spielstand speichern kann oder das Spiel beenden kann |

## Prozessdiagramm



# Lernjournal

## 22 Mai 2020

|  |  |  |
| --- | --- | --- |
| Zeit | Arbeitsbeschreibung | Beschreibung |
| 30min | Ideenfindung | Die Art unserer Applikation bestimmen, Auf MVVM oder MVC festlegen |
| 1h 15min | Konzeptionierung | Vorhandene Idee weiter ausarbeiten, Grundfunktionen unserer Applikation definieren |
| 15min | Planen, Vorbereiten | Gedanken zur Umsetzung, GitHub Repository und weitere Schritte |

Reflexion:

* Wir haben uns für eine MVVM Applikation entschieden. Wir tendieren sehr zu einem Game, welches simple Mechanismen aufweist. Als gute Idee kommt uns ein Idle Clicker oder ein sogenanntes Klicker Game vor.
* Wir habe genauere Details zum Game ausgearbeitet. Es soll “Hello World” heissen (Auch weil wir ja kein einfaches Hello World als Projekt wählen sollten), und das Ziel ist es so viel wie möglich “Hello World” auszugeben. Man kann sich Maschinen kaufen, die das “Hello World”-Ausgeben automatisieren.
* Wir haben uns vorgestellt mit einem GitHub Repository zu arbeiten, dieses haben wir auch erstellt. Ein Klassenkollege hat etwas von einem “Mark Down” erwähnt, und dass dieser die Dokumentation damit macht, eventuell möchten wir dem auch nachgehen.

## 27 Mai 2020

|  |  |  |
| --- | --- | --- |
| Zeit | Arbeitsbeschreibung | Beschreibung |
| 2h min | Konzept erweitern | Mockup und Prozessdiagramme erstellen |
| 1h 30min | Konzept verfeinern | Weitere Details abmachen, ausarbeiten |

Reflexion:

* Separat haben wir das Mockup für ein erstes Konzept des GUI’s mit Adobe XD entwickelt und ein Prozessdiagramm mithilfe von BPMN mit dem Bizagi Modeller erstellt.
* Unsere bisherigen Vorstellungen von der Applikation haben wir zusammengeführt und daraus ein noch feineres Konzept gemacht und über weitere Details entschieden. Es soll Upgrades geben, die die HW/s (“Hello World”’s pro sekunde) erhöhen (um bsp. 10%). Dann soll es Devices geben, die nach Netzwerk gruppiert sind, die HW/s produzieren. Die Devices kann man mit der Währung (Hello Worlds oder Karma welches man pro Hello World bekommt) kaufen. Neue Netzwerke oder Upgrades werden freigeschaltet, sobald man bestimmten Score von totalen HW/s übersteigt.
* Erste Ideen zu Klassen wurden geäussert.

## 28 Mai 2020

|  |  |  |
| --- | --- | --- |
| Zeit | Arbeitsbeschreibung | Beschreibung |
| 2h | Konzept erweitern | Mockup anhand von neuen Details erweitern/ändern |
| 2h | Prototyp entwickeln | Genäuer in MVVM und WPF einarbeiten, Informationen sammeln und einfache WPF-Applikation entwickeln |

Reflexion:

* Wir haben die Mockups nochmals überarbeitet, damit sie mit unseren Vorstellungen übereinstimmen.
* Wir versuchten uns an einem Prototyp und haben einfach mal ein WPF Projekt erstellt. Schliesslich konnten wir ziemlich einfach eine View erstellen, aber wie man dieses Mit dem ViewModel und dem Model verknüpft war uns komplett unklar. Diesen Prototyp verwarfen wir.

## 29 Mai 2020

|  |  |  |
| --- | --- | --- |
| Zeit | Arbeitsbeschreibung | Beschreibung |
|  | MVVM / WPF lernen | Tutorials anschauen und anhand von Programm testen |
|  |  |  |

Reflexion:

* Wir suchten diverse Tutorials durch, um das Prinzip von MVVM genauer zu verstehen.
* Den [Wikipedia-Artikel](https://de.wikipedia.org/wiki/Model_View_ViewModel#:~:text=Model%20View%20ViewModel%20(MVVM)%20ist,Logik%20der%20Benutzerschnittstelle%20(UI).) über MVVM sahen wir begeistert an, nach dem Lesen dieses Artikels war unsere Meinung, dass MVVM um einiges besser als MVC ist. Das auch weil im MVVM viel mehr geregelt und definiert ist und weil dieses Binding uns fasziniert hat.
* Wir haben auch nochmal in das [LinkedIn Tutorial](https://www.linkedin.com/learning/einfuhrung-in-die-softwarearchitektur-2-architekturmuster/was-ist-mvvm?u=2976210) reingeschaut
* Grundsätzlich ist das Prinzip von MVVM klar, aber codetechnisch haben wir noch nicht herausgefunden wie dies umgesetzt werden soll. Die Rede ist viel von Events, Delegates und so einem Interface namens INotifyPropertyChanged, all das kennen wir noch nicht.
* Ein [Youtube-Video](https://www.youtube.com/watch?v=EpGvqVtSYjs) erklärte das Prinzip von MVVM auch ziemlich gut, jedoch war uns immer noch nicht klar wie wir die Verknüpfung von Model ViewModel und View codetechnisch hinbringen sollten.

## 5 Juni 2020

|  |  |  |
| --- | --- | --- |
| Zeit | Arbeitsbeschreibung | Beschreibung |
| 30 | Versionskontrolle | VS-Projekt neu erstellt und auf github gepushed |
| 60 | Besprechung | Projekt besprochen und Änderungen dokumentiert |
| 60 | Events und Delegates | Tutorials zu Events und Delegates anschauen |

Reflexion:

* Vincent hat in seinem Betrieb MVVM angeschaut, und den Prototyp gemacht. Cedric hat unterdessen ein UML – Klassendiagramm – Prototyp gemacht, um unsere Ideen schon besser auf Papier zu bringen. Am Freitag 4.5 haben wir unsere Erfahrungen besprochen und Änderungen am Projekt vorgenommen. Ab nun haben wir kein Netzwerk mehr, da es das Projekt nur unnötig erschwert hätte. Ausserdem haben wir zusammen ein Neues UML -Klassendiagramm erstellt, welches das Projekt besser und einfacher erklärt. Wir haben auch das Nugget Fody besprochen (Beschreibung in diesem Dokument), welches das Programmieren erleichtert und den Code um einiges schlanker und übersichtlicher macht.
* Mit diesen Tutorials und Kommunikation zwischen uns, verstanden wir auch Events, und haben anhand von dem UML einen Verwendungszweck für diese gefunden ([Events und Delegates](https://youtu.be/jQgwEsJISy0?) / [Delegates](https://www.youtube.com/watch?v=G5R4C8BLEOc) / [Events](https://www.youtube.com/watch?v=TdiN18PR4zk))

# Tools und Quellen

## Fody (propertychanged)

Fody ist ein Nuggetpack welches sehr viele Erweiterungen für Visual Studio beinhaltet.  
Wir brauchen nur Fody propertychanged. Dieses Nugget generiert den Code nach dem er Gebuildet wird. Das heisst, er verändert die Intermediate Language. Ein Beispiel wäre die set-methode:

## Set-Methode mit Fody

### c#

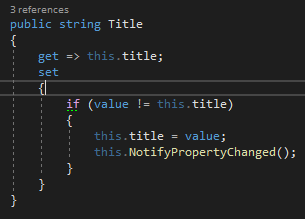


### Intermediate Language



## Set-MethodeOhne Fody

### c#



### Intermediate Language

