

WPF PC PART PICKER

Nicola Truscello IPT 2.1

Budget:

Usecase: Gaming

CPU

GPU

RAM

Storage

Name	Company	Cores	Socket	Company	Frequency	Freq. Box	Price
i3-10100	Intel	4	8	1200	3.6	4.3	100
i7-10700	Intel	8	16	1200	2.9	4.8	200
i3-13100	Intel	4	8	1700	3.4	4.3	
i5-10400	Intel	6	12	1200	2.9	4.3	
Ryzen 7 5800X	AMD	8	16	AM4	3.8	4.3	

ResultWindow

Ryzen 7 5800X	RTX 4090	Crucial Ballistix	WD B
i7-10700k	RX 6900 XT	Corsair ValueSelect	Samsu
i3-13100	RX 6900 XT	Corsair ValueSelect	Samsu
i3-13100	RTX 3070	Corsair ValueSelect	Samsu
i3-13100	RTX 3070	Trident Z Royal	Samsu
i3-13100	RTX 3070	Trident Z Royal	Crucia
i3-13100	RTX 3070	Trident Z Royal	Crucia
i5-10400k	RTX 3050	Trident Z Royal	Crucia
i3-10100	RTX 3050	Trident Z Royal	Crucia
i3-10100	RTX 3050	Trident Z Royal	Crucia
i3-10100	RTX 3050	Trident Z Royal	Crucia

Inhalt

Summary	3
Grobziele.....	3
Planung und Design.....	4
1.1 Grundlogik	4
3.1.1 Algorithmus	4
3.1.2 Importance	4
3.2 Einstellungen und Werte	5
3.2 Grunddesign	6
Benutzerhandbuch	7
Umsetzung und Tests.....	7
1. Algorithmus	7
1.1 Teil 1 - Sortierungsalgorithmus.....	7
1.2 Teil 2 - Bausystem	7
2. Tests.....	8
2.1 absolut nicht fähig → None Applicable & nur eine Generation → Funktioniert	8
2.2 bedingt fähig → None Applicable & nur eine Generation → Funktioniert.....	8
2.3 vollumfänglich fähig → Rest des Algorithmus → Funktioniert, aber nicht genügend Datensätze für diverse Beispiele	9
3. Hilfsmittel	9
Einträge.....	10
Eintrag 1 – 2. Woche - 05.04.2023.....	10
Eintrag 2 – Woche 4 - 03.05.2023.....	10
Eintrag 3 – Woche 8 - 14.06.2023.....	Erreur ! Signet non défini.
2. Lernjournal	10

Summary

Das Programm sollte mithilfe Eingaben des Users 10 passende PCs vorschlagen, die einem für einen neuen PC nützlich sein können.

Grundsätzlich lief das Projekt auch mit Verzögerungen gut. Obwohl mein Programm die Genauigkeit eines Experten nicht hat, konnte ich immerhin rechtzeitig ein Programm erstellen, die ihre Aufgabe gut erfüllt.

Die grösste Verzögerung war das OLE-DB Linking, welches scheinbar nicht mit den nötigen Login-Benutzerdaten, die man im Code angeben musste, funktionierte. Es funktionierte schlussendlich, ohne scheinbaren Grund. Immerhin merkte ich mir, dass diese Probleme auch grösser sein könnten und somit das ganze Projekt scheitern können.

In meinen nächsten Projekten würde ich derentwegen meine Erwartungen vertiefen und mehr Problem-Lösungszeiten in meinem Plan einbauen und erst nach dem Grundsätzlichen weiter expandieren.

Grobziele

Das Programm sollte mithilfe eines Budgets und Gebrauchsart passende PC-Setups vorstellen, damit der Benutzer ungefähr weiss, welche Hauptkomponente sie kaufen können. Das Programm stellt mehrere Beispiel-PCs vor, von einer Datenbank mit aktuellen Komponenten.

Planung und Design

1.1 Grundlogik

Mithilfe der Eingabe vom Budget und Gebrauchsart, sucht die Applikation bei der Datenbank entsprechende und passende Komponenten, die im Budget passen. Die Applikation schlägt zuerst die beste Kombination vor, bevor es andere Kombinationen vorschlägt, die Zufällig generiert werden. Maximal dürfen 10 Beispiele vorgeschlagen werden.

3.1.1 Algorithmus

- Jeder UseCase (Gebrauchsart) hat einen Prozentsatz für jedes Komponent, die zusammen alle 100% ergeben.

- Objekte für die individuelle Komponente werden so öfters instanziiert, wie es in der verlinkten Datenbank vorhanden sind.

- Sobald alles von der Datenbank abgerufen und in der internen Komponentenliste gespeichert ist, rechnet die Applikation die individuellen Budgets für alle Komponenten. Zunächst sortiert der Algorithmus alle unpassenden Komponenten weg und speichert die Kompatiblen in einer anderen Liste, die nach «Importance», bzw. die Wichtigkeit oder Stärke sortiert wird. Dieses «Importance» befindet sich in der Datenbank als eine weitere Spalte.

- Aus der Liste der kompatiblen Komponenten generiert die Applikation die beste Wahl zuerst. Danach generiert sie 6 weitere PCs, die Komponent nach Komponent die zweit-beste Wahl aussuchen. Zuletzt generiert sie die letzten 3 PCs, die entsprechend die dritt-besten Komponenten haben. Diesmal werden alle Komponenten den gleiche «Kompatibilität» haben. Dies bedeutet, dass zum Beispiel alle Komponente der 3. Letzten die dritt-beste Wahl ist.

Es werden also 10 PCs generiert.

3.1.2 Importance

Da eine neuere Generation nicht unbedingt ein besseres Produkt andeutet, werden alle Tabellen der Datenbank mit "Importance" erweitert.

Das leistungsschwächste Produkt der Komponententabelle fängt mit der 1 an. In ihrer eigenen Generation inkrementiert es bis zu ihrem Flagship. Bei dem nächsten Gen. fängt das leistungsschwächste Produkt des Line-ups mit der Zahl des zweit-schlechtesten Produktes der letzten Generation an und inkrementiert sich weiter.

Andere Komponenten, wie RAM, können nicht so einfach in dieser Sortierungsart passen und werden nach technologischer Fähigkeit und Leistungszahlen inkrementiert:

Für den RAM wird nach Megatransfer schritt von 2133MHz +1 gerechnet. Je nach Verdopplung der Grösse von 8GB, wird +2 gerechnet. Je nach Generation wird +2 gerechnet.

Für Storage wird nach Interface, Grösse und Typ sortiert.

NVMe Protokolle fügen +3, normale SSDs +2, HDDs nichts.

Je nach Grössenschritt von 128 wird exponentiell von 128 gerechnet.

Also fügt 128GB nichts, 256GB +1, 500GB +2, 1TB +4, etc.... Der Bottom-Line ist 128GB HDD.

Es kann aber auch sein, dass der Generationale fortschritt schwach ist und somit den Importance nicht wie gewöhnlich ändert.

3.2 Einstellungen und Werte

DATABASE COLUMNS:

Columns for CPU (MAX 3 GEN BACK): FullName; Cores; Threads; Frequency; FrequencyBoost; Company; Generation; Cost

Columns for GPU (MAX 3 GEN BACK): FullName; Company; Generation; Cost

Columns for RAM: FullName; Size; MT/s; Latency; Company; DDR Gen; Cost

Columns for STOR: FullName; Company; Type; FormFactor; Size; Cost

Columns for UseCase: FullName; CPUPercent; GPUPercent; RAMPercent; STORPercent

Columns for STORAUX: FullName; Company; Type; FormFactor; Size; Cost

USECASES:

Office: (CPU 30% GPU 0% RAM 20% STORAGE 50%)

Gaming: (CPU 25% GPU 40% RAM 15% STORAGE 20%)

3D Modelling: (CPU 30% GPU 30% RAM 25% STORAGE 15%)

Mixing: (CPU 30% GPU 0% RAM 30% STORAGE 40%)

CAD: (CPU 20% GPU 30% RAM 20% STORAGE 30%)

AI: (CPU 25% GPU 65% RAM 10% STORAGE 05%)

Video man.: (CPU 25% GPU 25% RAM 20% STORAGE 25%)

Mining: (CPU 05% GPU 85% RAM 05% STORAGE 05%)

3.2 Grunddesign

- Bei UseCase wird das Nutzen des PCs beschrieben.
- Bei Budget wird das Budget eingegeben.
- Die Tabelle und Überordner zeigt eine Liste aller gespeicherten Produkte in ihre Komponentenkategorie auf. Bei einem Drücken der Überordner wechselt sich die Darstellung entsprechend den Komponenten.
- Die quadratische Darstellung, die rechts zu sehen ist, zeigt die Aufteilung des Budgets zu den entsprechenden Komponenten auf.
- Es soll unterhalb der Darstellung rechts ein «Generate» Knopf hinzugefügt werden.

ID	CPU	GPU	RAM	STOR	AUX STOR	COST

- Alle Beispiel-PCs werden bei dieser Tabelle aufgezeigt.

Benutzerhandbuch

(Separates Dokument)

Umsetzung und Tests

1. Algorithmus

Der Algorithmus besteht aus 2 Hauptteile, die relativ einfach aufgebaut sind, aber schlussendlich funktionieren.

Da der Rest des Programms entweder Events oder Datenbankaufrufe beinhaltet und nur der Algorithmus wesentlich komplizierter ist, steht der Algorithmus im Focus.

1.1 Teil 1- Sortierungsalgorithmus

Das Sortierungsalgorithmus sortiert mit einem Durchgang von der ursprünglichen Liste mit allen Datensätze die passende Komponente aus. Diese aussortierten Komponenten, die wie gesagt aussortiert werden, werden dann nach den «Importance» sortiert. Das passiert mit einem Bubble-sort. Es vergleicht mit einer For-schleife und einem If-statement, um das nächste Komponent der Liste mit den ersten zu vergleichen. Falls die Bedingung vom innerlichen If-statement gültig wird, tauschen beide Komponenten ihre Plätze.

Eine zweite For-schleife mit einem If-statement überprüft diesmal, ob das erste indexierte Komponent das gleiche Importance wie der zweite hat und ob das zweite günstiger ist und tauscht somit wie im ersten Loop ihre Plätze, um das günstigste vorzuholen.

Jedes Komponent hat genau dieselben 2 Schleifen, um das Prozess für alle Komponenten zu durchführe.

1.2 Teil 2- Bausystem

Das Bausystem wird nach der Sortierung aufgerufen. Zuerst wird überprüft, ob es bei jeder sortierten Liste keine passenden Komponenten gibt. Falls dies der Fall ist, erstellt das Programm ein neues Komponent entsprechend der leeren Liste und nennt es «None Applicable».

Somit kann das System ein PC bauen, die die besten Exemplare hat. Lücken werden, wie vorher erläutert, mit einem Placeholder gefüllt.

Das Programm schaut danach, ob die Summe aller Längen der Sortierten liste über 12 ist. Falls das nicht der Fall ist, hört das Programm mit einem Exemplar auf.

Im Gegenteil macht es mit ihrem Algorithmus weiter.

Mithilfe einer For-schleife und einem Switch-case erstellt das Programm einen weiteren PC, indem der Switch-case je nach Index der For-schleife den Index einer von den 5 Komponentenlisten erhöht.

Also nimmt das Programm beispielsweise den zweitbesten CPU und die besten restlichen Komponenten für den zweiten PC, danach dann zweitbesten CPU und GPU und die besten restlichen Komponenten für das dritte, usw.

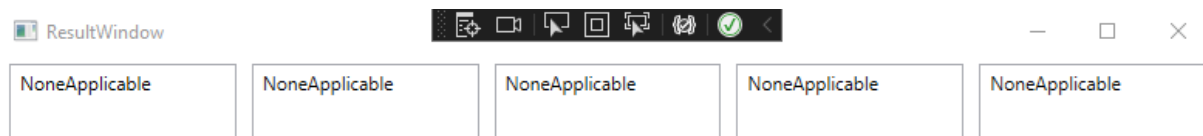
Somit werden 6 weitere PCs erstellt. Um die restlichen 3 zu generieren, tritt der «Default» case im Switch-case ein und erhöht, denn Index von allen Komponentenlisten gleichzeitig, damit der nächste PC alles nur die dritt-besten, die viert-besten und fünft-besten entsprechend nimmt.

2. Tests

Um die beschriebenen Units zu testen, wurden folgende PCs generiert mit folgenden Eigenschaften und Resultate.

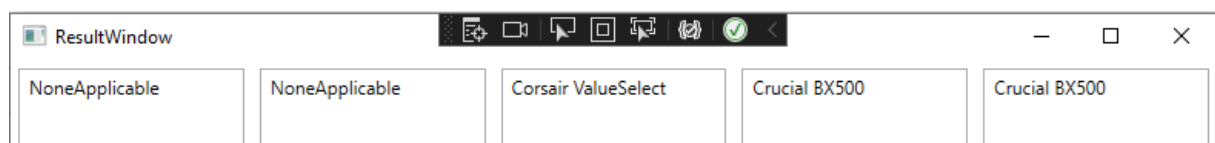
2.1 absolut nicht fähig → None Applicable & nur eine Generation → funktioniert

Budget: 0, Usecase: Gaming



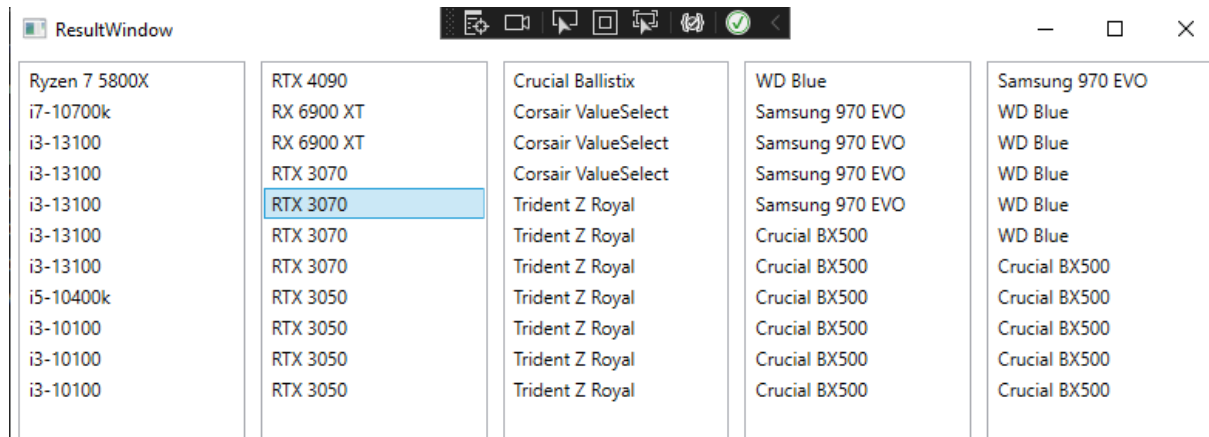
2.2 bedingt fähig → None Applicable & nur eine Generation → funktioniert

Budget: 100, Usecase: Gaming



2.3 vollumfänglich fähig → Rest des Algorithmus → funktioniert, aber nicht genügend Datensätze für diverse Beispiele

Budget: 1231231, Usecase: Gaming



ResultWindow				
Ryzen 7 5800X	RTX 4090	Crucial Ballistix	WD Blue	Samsung 970 EVO
i7-10700k	RX 6900 XT	Corsair ValueSelect	Samsung 970 EVO	WD Blue
i3-13100	RX 6900 XT	Corsair ValueSelect	Samsung 970 EVO	WD Blue
i3-13100	RTX 3070	Corsair ValueSelect	Samsung 970 EVO	WD Blue
i3-13100	RTX 3070	Trident Z Royal	Samsung 970 EVO	WD Blue
i3-13100	RTX 3070	Trident Z Royal	Crucial BX500	WD Blue
i3-13100	RTX 3070	Trident Z Royal	Crucial BX500	Crucial BX500
i5-10400k	RTX 3050	Trident Z Royal	Crucial BX500	Crucial BX500
i3-10100	RTX 3050	Trident Z Royal	Crucial BX500	Crucial BX500
i3-10100	RTX 3050	Trident Z Royal	Crucial BX500	Crucial BX500
i3-10100	RTX 3050	Trident Z Royal	Crucial BX500	Crucial BX500

3. Hilfsmittel, Tutorials und Einarbeitung

Wie es am Anfang geplant war, wollte ich mithilfe Web-APIs meine Beispiel-PCs konstruieren. Dementsprechend habe ich passende Tutorials geschaut, die mich grundsätzlich dazu einführen. Dies wurde aber abgebrochen, da die grössten Webshops mit guten Kategorien und Einteilung der Produkte entweder keine APIs hatten (wie Digitec), oder kostenpflichtige APIs, die nur an Partner vergeben werden.

Immerhin waren die ersten Videos der Playlist interessante Einstiege in APIs.

- https://www.youtube.com/playlist?list=PLhW3qG5bs-L-aUxyATNOB_L5Qk-a5TfM-

Da ich nach der Ablehnung von Digitec und Lizenzen nicht mit APIs arbeiten konnte, entschied ich mit Datenbanken zu arbeiten, mit Access und OLE-DB, um genauer zu sein.

Mit den Datenbanken wurden keine Tutorials gebraucht, aber es wurde mithilfe einer Aufgabe von der Lehrperson erlernt.

Andere, kleinere Details wurden mithilfe des Internets in verschiedene Forums gelöst und brauchten daher keine Tutorials.

Einträge

Eintrag 1 – 2. Woche - 05.04.2023

Das PC-Part-Picker-Programm war geplant, Produkte sofort von einer Webseite, wie Digitec, abzufragen. Das soll dazu dienen, echte Produkte vorzuschlagen, anstatt nur Spezifikationen und ungefähre Preise. Um zu arbeiten, habe ich die Tutorials entsprechend dieser Playlist angeschaut: «https://www.youtube.com/playlist?list=PLhW3qG5bs-L-aUxyATNOB_L5Qk-a5TfM». Dazu wurden viele Online-Shops aufgerufen, um entsprechende API's zu finden.

Leider sind die API's der grössten Online-Shops noch nicht entwickelt worden, oder sind hinter einer Lizenz verborgen, welches man erst haben kann, wenn man ein Partner ist.

Als eine Alternative wird entweder XML oder DBs gebraucht, je nachdem wie weit man kommt.

<https://learn.microsoft.com/en-us/visualstudio/data-tools/connect-to-data-in-an-access-database-windows-forms?view=vs-2022>

Eintrag 2 – Woche 4 - 03.05.2023

Nach langem Troubleshooting funktioniert die Datenbanken-Verlinkung. Es wird mit einer Datenbank weitergefahren.

2. Lernjournal

WOCHE	SOLL	IST +	IST -
-1 (START)	-Dokumentation anfangen, ggf. was möglich ist fertigstellen. -Restliche Lektion	Dokumentationsgrundlagen erfüllt, bzw. die Dokumentationsstruktur und die Grobplanung (Schule)	
1. -29.03.2023	-Einführung Webservice & Einarbeitung - >3 Lektionen Haupt GUI - 1 Lektion	Webservice recherchiert, Tutorials angeschaut. (Schule) Digitec für Zugang nach internen API nachgefragt. (Zuhause) Haupt GUI Grundlagen entwickelt, Inputkasten und alle nötigen Grids erstellt (Linker und rechter Grid, Darstellungsgrid für die Aufteilung der Komponenten im Budget) (Schule)	

2. -05.04.2023	Je nach dem Webservice oder DB-Verknüpfung lernen.	DB mithilfe der Lehrperson und Rollerapplikationsaufgabe erlernt, DB mit Angaben erstellt. (Während Homeschooling)	<p>Webservice-idee gestrichen, da Ablehnung von Digitec und Lizenzpflicht anderer Webshops.</p> <p>Die Verlinkung wurde mithilfe der Rollerapplikation versucht, indem man den Code kopiert und an der DB anpasst, funktionierte aber nicht. (zu Hause zuzüglich 2 Stunden)</p>
3. -25.04.2023	<p>DB-Weiterentwickeln, soweit kommen, dass es für min. 1. Obj. funktioniert.</p> <p>DB-Linking implementieren. (bzw. fragen, wieso nicht funktioniert)</p>	<p>DB list-auffüllung wurde für alle Objekten, bzw. Komponenten gemacht. (Schule)</p> <p>In der Zwischenzeit das Haupt GUI Struktur übersichtlicher gemacht mit Abständen, Kommentars und Verbesserungen zu nicht gültigen Containern.</p>	DB Linking funktioniert immer noch nicht, aber mit der Lehrperson angeschaut.
4. -03.05.2023	DB-Linking fertigstellen, Auffüllung fertigstellen	<p>DB-Linking funktioniert vollumfänglich</p> <p>Objekte und die Auffüllungsfunktionen wurden fertiggestellt, funktionieren. (Schule)</p> <p>Das Haupt GUI wurde weiterhin erkundet und teilweise erlernt mit ListBox und CompoundBox. (Schule)</p>	
5. -10.05.2023	<p>Haupt GUI erstellen, anfangen.</p> <p>DB fertigstellen, Auffüllung fertigstellen</p>	DB fertiggestellt (Alle Tabellen erstellt und testobjekte eingefügt) und erster Teil des Algorithmus fertiggestellt. (Sortierung nach Importance)	Haupt GUI nach Versuch der Implementation vom «Combobox» zur Seite gestellt. Die «SelectionChanged» Event wechselt nach Selektionswechsel die eigentlich zu

			darstellen Komponente nicht.
6. -17.05.2023	Haupt GUI anfangen, Algorithmus fertigstellen.	Algorithmus verbessert, Sortierung mit Preis und nicht nur Importance. (Schule & zuhause, zuzüglich 1 Stunde) Haupt GUI mit ComboBox weitergeführt. Listing funktioniert nicht komplett, aber grundsätzlich. (Schule)	Algorithmus wurde nach
7. -24.05.2023	Resultats Fenster erstellen, Algorithmus weiterfahren, damit es richtig die Komponenten sortiert, mit korrekten Kombinationen baut und schlussendlich alles im Resultats Fenster anzeigt	Git wurde eingeführt, bzw. Branches, Commits, etc. (Schule) Listboxes beim Haupt GUI weiterentwickelt für andere Komponenten. (Zuhause) Resultatfenster erstellt, mit Code, um die Listboxes zu füllen und getestet (zuhause, zuzüglich 2 Stunden) Generate-Button und code dahinter anfangen. (Algorithmus Aufruf, Sortierung Komponenten, PC bauen gemäss Grobplanung) (Schule & zuhause, zuzüglich 1 Stunde)	
8. -07.06.2023	Mit dem Algorithmus und mit dem Programm generell weiterfahren und folglich fertigstellen.	Algorithmus verbessert, indem es nicht zufälligerweise Komponenten aussucht und systematisch fortfahrt, da die Zeit kein Problem mehr ist. (Schule und zuhause, zuzüglich 3 Stunden)	
9. -14.06.2023	Dokumentation fertigstellen und letzte	Der Algorithmus wurde richtig implementiert und für Übersichtlichkeit die	

	<p>Verschönerungen eingehen.</p> <p>Das Programm, das Algorithmus (Speziell die letzten 3 zufälligen Beispiel-PCs) fertig implementieren.</p> <p>Video über das Programm erstellen.</p> <p>Projekt abgeben.</p>	<p>Variabel Namen abgeändert für Konsistenz. (zu Hause, zuzüglich 2,5 Stunden)</p> <p>Video erstellt und bearbeitet (zu Hause, zuzüglich 1,5 Stunden)</p> <p>Das Summary und Umsetzung und Tests geschrieben, folglich Dokumentation fertigerstellt.</p>	
--	---	--	--