HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

Unser tolles Thema – wir sind genial

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:
Robert Freiseisen
Philipp Füreder

Betreuer:
Rainer Stropek

Leonding, August 2023

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022 S. Schwammal & S. Schwammal

Abstract

Brief summary of our amazing work. In English. This is the only time we have to include a picture within the text. The picture should somehow represent your thesis. This is untypical for scientific work but required by the powers that are. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Zusammenfassung

Zusammenfassung unserer genialen Arbeit. Auf Deutsch. Das ist das einzige Mal, dass eine Grafik in den Textfluss eingebunden wird. Die gewählte Grafik soll irgendwie eure Arbeit repräsentieren. Das ist ungewöhnlich für eine wissenschaftliche Arbeit aber eine Anforderung der Obrigkeit. *Bitte auf keinen Fall mit der Zusammenfassung verwechseln, die den Abschluss der Arbeit bildet!* Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Inhaltsverzeichnis

1 Einleitung

1.1 Standardsoftware vs Individualsoftware

In der heutigen digitalisierten Welt ist Software enorm wichtig für den Erfolg von Unternehmen oder auch für den privaten Gebrauch. Es gibt zwei Hauptarten von Software, die von Unternehmen genutzt werden: **Standardsoftware** und **Individualsoftware**. Beide haben Vor- und Nachteile, und es ist wichtig, diese gut zu verstehen, um die richtige Wahl für das eigene Unternehmen zu treffen.

Standardsoftware

Standardsoftware ist üblicherweise sehr Benutzerfreundlich und einfach zu verwenden. Sie wird meist von großen Unternehmen entwickelt (Beispiel: Microsoft Office, Adobe Creative Cloud oder auch Google Workspace) und ist in der Anschaffung billiger als Individualsoftware. Der Grund dafür ist, dass Standardsoftware für die breite Masse entwickelt wird und für jeden Nutzer die gleichen Funktionen besitzt. Außerdem sind sie in der Regel weiter verbreitet und leichter zu erwerben.

Ein Nachteil von Standardsoftware besteht darin, dass sie nicht immer genau den individuellen Anforderungen eines Unternehmens gerecht wird. Es könnte vorkommen, dass spezifische Funktionen fehlen oder die Software nicht optimal auf die Arbeitsprozesse des Unternehmens zugeschnitten ist.

Individualsoftware

Auf der anderen Seite wird Individualsoftware speziell für ein einzelnes Unternehmen oder eine bestimmte Aufgabe entworfen. Das bedeutet, es ist eine individuelle Lösung, die exakt auf die Anforderungen und Bedürfnisse der Benutzer/innen zugeschnitten ist.

Die Verwendung von Individualsoftware hat aber auch ihre Nachteile. Zum einen ist sie in der Regel kostenintensiver, da die individuelle Entwicklung mehr Ressourcen erfordert. Zum anderen gestaltet sich die Umsetzung und Wartung anspruchsvoller, da die Einzigartigkeit der Software spezifische Herausforderungen mit sich bringt, die über die Zeit hinweg bewältigt werden müssen.

1.2 Problemverständnis

Die meisten Benutzer/innen sind mit den gängigsten Standardsoftwares bereits vertraut. Deshalb wäre eine neue Individualsoftware eine Umstellung, wo man wieder Zeit aufwenden muss, um sich damit zurecht zu finden. Und außerdem muss man diese dann ja auch implementieren (als Entwickler), warten und vor allem finanzieren, was sehr kostspielig sein kann.
Es mag verlockend sein, Standardsoftware mit den benötigten Funktionen für jeden Kunden anzupassen. Aber das bringt viele Probleme mit sich. Wenn man für jeden Kunden spezielle Funktionen einbaut, muss man den Software-Code jedes Mal stark ändern. Das macht die Software kompliziert und schwer zu pflegen. Es können auch Fehler auftreten, wenn neue Funktionen hinzugefügt werden, und es wird schwierig, die Software auf dem neuesten Stand zu halten.

1.3 Lösungsansatz: Scripting

Ein vielversprechender Ansatz zur Lösung von fehlenden Funktionen in einer Software besteht darin Scripting zu nutzen. Dadurch erhalten Benutzer/innen die Möglichkeit, eigene Scripts zu erstellen, um spezifische Funktionen innerhalb der Software auszuführen. Diese Scripts werden zur Laufzeit in die Anwendung geladen.

Allgemeines zu Scripting

Scripting ermöglicht es individuelle Lösungen zu erstellen, die den eigenen Anforderungen gerecht werden. Dies erweitert den Nutzen der Software, ohne auf offizielle Updates oder neue Versionen warten zu müssen oder sogar eine eigene Software entwickeln zu müssen.

Eine Skriptsprache wird vor allem verwendet, um Websites und Webanwendungen zu automatisieren. Wenn man ein Skript schreibt, baut man kein völlig neues Programm von Grund auf. Stattdessen verknüpft man bestehende Teile eines Programms miteinander. Dann führt das Programm dieses Skript aus.

Ein zentraler Aspekt dieser Diplomarbeit war die Untersuchung der Rolle des Scripting bei der dynamischen Anpassung von .NET-Anwendungen. Der Einsatz von Skriptsprachen ermöglicht es, fehlende oder zusätzliche Funktionen zur Laufzeit in eine Anwendung zu integrieren. Im Rahmen unserer Arbeit wurde verschiedenen Skriptsprachen experimentiert, um deren Eignung für die Integration in .NET-Anwendungen zu bewerten. Insbesondere wurde erforscht, wie Skripte nahtlos in die .NET-Anwendungen eingebettet und ausgeführt werden können, um die Flexibilität und Erweiterbarkeit der Anwendungen zu erhöhen.

Vorteile

- **Schnelle Entwicklung:** Skriptsprachen werden interpretiert, wodurch Code rasch entwickelt und getestet werden kann. Dies ermöglicht ein einfaches Ausprobieren neuer Ideen und die zügige Erzielung von Ergebnissen, was Skriptsprachen besonders für interaktive Programmieransätze geeignet macht.
- **Benutzerfreundlichkeit:** Skriptsprachen zeichnen sich durch ihre Benutzerfreundlichkeit aus, wobei der Fokus auf der simplen und intuitiven Gestaltung gängiger Aufgaben liegt. Diese Eigenschaft eignet sich perfekt für Anfänger sowie für Aufgaben, die keine umfangreiche Rechenleistung erfordern.
- **Flexibles Verhalten:** Häufig weisen Skriptsprachen eine dynamische Typisierung auf, die es ermöglicht, dass der Typ einer Variablen während der Laufzeit verändert werden kann. Dies vereinfacht die Erstellung flexiblen und anpassungsfähigen Codes sowie den Umgang mit Daten in vielfältigen Formaten.

Nachteile

- Der größte Nachteil von Skriptsprachen ist, dass sie langsamer sind als andere Programmiersprachen. Das liegt daran, dass in Skriptsprachen jedes Statement nacheinander während der Ausführung gelesen und verarbeitet wird.
- Wenn während der Ausführung des Skripts ein Fehler entdeckt wird, stoppt der Interpreter die Ausführung, bis dieser korrigiert wird.

Levels von Scripting

In Microsoft Excel gibt es die Möglichkeit sich wiederholende Aufgaben mithilfe von Office Scripts zu automatisieren.
Man kann auf zwei Arten ein neues Office-Skript erstellen:

- Man kann seine Handlungen mithilfe des Actionrecorders aufnehmen. Diese Methode eignet sich besonders gut, wenn man sich wiederholende Schritte in dem Dokument merken möchte. Dazu sind keine Programmierkenntnisse oder ähnliches erforderlich. Die Aufgezeichneten Skripte können abgespeichert und verändert werden.
- Die zweite Möglichkeit ist, dass Office-Skript selbst mithilfe von TypeScript zu schreiben.

Folgendes Office-Skript Beispiel gibt den Wert von der Zelle A1 auf der Konsole aus:

Listing 1: Office-Skript

```
1     function main(workbook: ExcelScript.Workbook) {
2   // Get the current worksheet.
3   let selectedSheet = workbook.getActiveWorksheet();
4
5   // Get the value of cell A1.
6   let range = selectedSheet.getRange("A1");
7
8   // Print the value of A1.
9   console.log(range.getValue());
10  }
```

1.4 Alternativen

Microsoft Power Automate

Power Automate ist eine cloudbasierte Plattform, die es Anwender/innen unkompliziert ermöglicht, Workflows zu erstellen. Diese Arbeitsabläufe automatisieren zeitaufwändige geschäftliche Aufgaben und Prozesse, indem sie Anwendungen und Dienste verbinden.

Logik-Apps (ein Service von Azure), präsentiert vergleichbare Eigenschaften wie Power Automate. Zusätzlich dazu bietet es weitere Leistungsmerkmale, darunter die nahtlose Einbindung in den Azure Resource Manager, das Azure-Portal, PowerShell, die xPlat-CLI, Visual Studio und diverse weitere Verbindungselemente.

Mit folgenden Dienstleistungen können Power Automate verbunden werden:

- Sharepoint
- Dynamics 365
- OneDrive
- Google Drive
- Google Sheets
- und noch einige mehr

Power Automate ist außerdem plattformübergreifend und kann auf allen modernen Geräten und Browsern ausgeführt werden. Zur Verwendung ist nur ein Webbrowser und eine E-Mail-Adresse erforderlich.

Dynamische Modulsysteme

Ein Ansatz, der oft angewandt wird, um Anwendungen oder Systeme zu konstruieren, ist die Nutzung eines dynamischen Modulsystems. Diese Methode ermöglicht die Erstellung von Applikationen durch den Zusammenbau wiederverwendbarer Einzelmodule. Diese Herangehensweise erleichtert die Entwicklung komplexer Systeme, ohne dass jedes Mal von Grund auf ein völlig neues System erstellt werden muss. Ein weiterer Nutzen besteht darin, dass verschiedene Anwendungen oder Systeme auf diese Weise miteinander verknüpft werden können. Modulsysteme werden oft in der Softwareentwicklung, Webentwicklung, Datenbanken und Systemadministration verwendet.

Beispiele für Modulsysteme:

Webanwendung

- Javascript-Frameworks wie React und Angular

Desktop- und Serveranwendung

- .NET und Java

Erstellung und Verwaltung von Datenbanken

- PostgreSQL und MongoDB

Verwaltung von Netzwerken und Systemen

- Puppet und Chef

Microservices

Microservices stellen einen Ansatz in der Softwareentwicklung dar. Er basiert darauf, dass Software aus einer Vielzahl kleiner, eigenständiger Dienste besteht. Diese Dienste interagieren miteinander über klar definierte Schnittstellen. Die Architektur von Microservices trägt dazu bei, Skalierbarkeit zu erleichtern und die Zeitspanne für die Entwicklung von Anwendungen zu verkürzen. Dies ermöglicht eine schnellere Umsetzung von Innovationen und beschleunigt die Einführung neuer Funktionen auf dem Markt.

Eigenschaften von Microservices

Eigenständigkeit:
In einer Architektur von Mikroservices besteht die Möglichkeit, jeden einzelnen Komponentenservice unabhängig zu entwickeln, bereitzustellen, zu betreiben und zu skalieren. Hierbei hat die Veränderung eines Services keine Auswirkungen auf die Funktionalität anderer Services. Es ist nicht erforderlich, dass Services Code oder Implementierungen miteinander teilen. Die Interaktion zwischen den verschiedenen Komponenten erfolgt ausschließlich über eindeutig definierte APIs.

Spezialisierung:
Jeder einzelne Service ist darauf ausgerichtet, eine spezifische Gruppe von Problemstellungen zu lösen. Sollte man im Laufe der Zeit zusätzlichen Code zu einem solchen Dienst hinzufügen, und dadurch der Service zu komplex wird, besteht die Option, diesen in kleinere Services aufzuteilen.

Dynamisches Laden von Assemblies

Das .NET-Framework biete die Möglichkeit, Assembly-Dateien zur Laufzeit zu laden. Dies ermöglicht es, neue Funktionen in Form von Klassen und Methoden hinzuzufügen. Assemblies stellen die Grundbausteine von .NET-Anwendungen dar und treten in Form von ausführbaren Dateien (.exe) oder Dynamic Link Library-Dateien (.dll) auf.

Folgender Beispielcode dient dazu, eine Assembly mit der Bezeichnung "example.exe" innerhalb der aktuellen Anwendungsdomäne zu laden. Anschließend wird ein Typ mit dem Namen "Example" aus dieser Assembly abgerufen. Auf diesen abgerufenen Typ wird die Methode "MethodA" ausgeführt.

Listing 2: Assembly

```
1       using System;
2       using System.Reflection;
3
4       public class Asmload0
5       {
6           public static void Main()
7           {
8               // Use the file name to load the assembly into the current
9               // application domain.
10              Assembly a = Assembly.Load("example");
11              // Get the type to use.
12              Type myType = a.GetType("Example");
13              // Get the method to call.
14              MethodInfo myMethod = myType.GetMethod("MethodA");
15              // Create an instance.
16              object obj = Activator.CreateInstance(myType);
17              // Execute the method.
18              myMethod.Invoke(obj, null);
19          }
20      }
```

1.5 Beschreibung über die Durchführung der Diplomarbeit

1.6 Machbarkeitsnachweis (Beispielanwendung)

2 Umfeldanalyse

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. Citing [1] properly.

Was ist eine GUID? Eine GUID kollidiert nicht gerne.

Kabellose Technologien sind in abgelegenen Gebieten wichtig [2].

3 Anwendung (Praxisteil)

3.1 Verwendete Technologien

ASP .NET Core

Unsere Webanwendung wurde unter Verwendung der ASP.NET Core-Plattform entwickelt. ASP.NET Core ist ein vielseitiges, plattformübergreifendes und leistungsfähiges Open-Source-Framework, das zur Entwicklung moderner, internetfähigen Anwendungen geeignet ist. Die Ausführung findet in der .NET Core-Laufzeitumgebung statt. ASP.NET Core bietet außerdem eine moderne und flexible Umgebung für die Entwicklung von Webanwendungen, die sowohl plattformübergreifend als auch hochgradig skalierbar sind.

Mit ASP.NET Core kann man:

- Webanwendungen und Webdienste, Internet-der-Dinge (IoT)-Anwendungen und mobile Backends entwickeln.
- Auf verschiedene Betriebssysteme wie Windows, macOS und Linux arbeiten.
- Anwendungen sowohl in der Cloud als auch auf lokalen Systemen bereitstellen.

Dieses Framework eröffnet somit eine breite Palette an Möglichkeiten für Entwickler, um moderne Anwendungen zu erstellen, die sich nahtlos mit dem Internet verbinden und sowohl in Cloud- als auch lokalen Umgebungen effizient betrieben werden können.

Git

Unsere Versionskontrolle haben wir mit Git gemacht. Git ist ein Versionskontrollsystem mit verteiltem Ansatz, das entworfen wurde, um sowohl kleine als auch äußerst umfangreiche Projekte auf schnelle und effiziente Weise zu verwalten. Die Erlernbarkeit von Git gestaltet sich einfach, und seine geringe Systembelastung geht einher mit herausragender Performance. Es setzt sich von anderen Versionskontrollsystemen wie Subversion, CVS, Perforce und ClearCase ab, indem es Funktionen wie kosteneffiziente lokale „Branches“, bequeme Staging-Bereiche und vielfältige Arbeitsabläufe bietet. Git erlaubt und begünstigt die Erstellung von mehreren unabhängigen lokalen Verzweigungen. Die Prozesse des Erstellens, Zusammenführens und Entfernens dieser Entwicklungsstränge nehmen lediglich Sekunden in Anspruch.

Git ist kostenfrei und Open-Source. Es wurde dazu entwickelt, Projekte aller Größenordnungen – von kleinen bis hin zu umfangreichen – schnell und effizient zu verwalten. Es zeichnet sich als Open-Source aus, da es die Anpassungsfähigkeit bietet, den Quellcode nach den individuellen Bedürfnissen der Nutzer/innen anzupassen. Mit seiner Open-Source-Natur ermöglicht Git mehreren Personen gleichzeitig an einem Projekt zu arbeiten und ermöglicht eine äußerst einfache und effiziente Zusammenarbeit. Aus diesem Grund wird Git als das herausragende Versionskontrollsystem betrachtet, das in der heutigen Zeit zur Verfügung steht.

Docker

Docker ist eine Plattform, welche Anwendungen gemeinsam mit ihren spezifischen Abhängigkeiten in Form von Containern bündelt. Dieser Ansatz gewährleistet, dass die Anwendung in jeder beliebigen Entwicklungsumgebung reibungslos funktioniert. Jede einzelne Anwendung läuft in separaten Containern und verfügt über ihre eigenen Satz an Abhängigkeiten und Bibliotheken. Dies gewährleistet, dass jede Applikation in völliger Unabhängigkeit von anderen Anwendungen agiert und die Sicherheit gibt, dass sie Anwendungen erstellen können, welche sich nicht gegenseitig beeinträchtigen. Wir haben in unserer Anwendung unser Frontend, Backend und die Datenbank über Docker laufen lassen. Das heißt wir haben 3 Docker-Container benutzt.

Unter einem Container versteht man eine standardisierte Softwareeinheit, die sowohl den Programmcode als auch sämtliche damit verbundenen Abhängigkeiten zusammenfasst. Hierdurch wird sichergestellt, dass die Anwendung zuverlässig in unterschiedlichen Computerumgebungen ausgeführt werden kann. Ein Docker-Container-Image verkörpert eine autonome und ausführbare Softwareeinheit, welche sämtliche Komponenten für die Ausführung einer Applikation in sich trägt: den Code selbst, die Laufzeitumgebung, Systemwerkzeuge, Systembibliotheken und Konfigurationseinstellungen.

Container-Images verwandeln sich zur Laufzeit in eigenständige Container. Im Fall von Docker geschieht dies durch das Ausführen der Images auf der Docker Engine. Containerisierte Software steht sowohl für Linux- als auch für Windows-basierte Anwendungen zur Verfügung und gewährleistet eine gleichbleibende Ausführung, unabhängig von der genutzten Infrastruktur. Container schaffen eine Isolierung der Software von ihrer Umgebung und gewährleisten somit, dass die Anwendung konsistent arbeitet, selbst bei Unterschieden zwischen Entwicklungs- und Staging-Umgebungen.

PostgreSQL

Als Datenbanksystem haben wir uns für PostgreSQL entschieden, da wir bereits in anderen Projekten mit diesem gearbeitet haben. PostgreSQL ist Open-Source und verwendet die SQL-Sprache. Außerdem ist PostgreSQL auf allen gängigen Betriebssystemen kompatibel. PostgreSQL läuft bei uns über Docker, somit haben wir nichts Zusätzliches dafür installieren müssen. PostgreSQL ist sehr anpassbar, man kann beispielsweise eigene Datentypen definieren und individuelle Funktionen gestalten.

Datentypen in PostgreSQL:

- Zahlen und Zeichen: Integer, Numeric, String, Boolean
- Strukturierte: Date/Time, Array, Range/Multirange
- Geometrisch: Point, Line, Circle, Polygon
- Anpassbare: Composite, Custom Types

Integrität der Daten:

- UNIQUE, NOT NULL
- Primary Keys
- Foreign Keys
- Exclusion Constraints
- Explicit Locks, Advisory Locks

BenchmarkDotNet (0.13.2)

BenchmarkDotNet unterstützt Anwender/innen dabei, die Performance ihrer Methoden in Benchmark-Tests zu überprüfen. Ebenso ermöglicht es den Austausch von reproduzierbaren Messexperimenten. Diese Transformation gestaltet sich genauso unkompliziert wie die Erstellung von Unit-Tests. BenchmarkDotNet hilft dabei, übliche Fehler im Benchmarking-Prozess zu vermeiden und Nutzer zu informieren, sobald Unstimmigkeiten im Benchmark-Design oder den erfassten Messdaten auftreten. Die präsentierten Resultate erscheinen in einer nutzerfreundlichen Tabelle, die sämtliche relevanten Aspekte des Experiments herausstellt.
Anbei ein Beispiel von einem unserer BenchmarkDotNet-Tests:

Listing 3: BenchmarkDotNet

```
1  namespace C_SharpExample
2  {
3      [MarkdownExporter,
4          HtmlExporter,
5          SimpleJob(RunStrategy.ColdStart, launchCount: 1, warmupCount: 5,
               targetCount: 5, id: "FastAndDirtyJob")]
6      public class C_SharpTesting
7      {
8          [Benchmark]
9          public void TestC_Sharp_Simple() => ReturnNumber();
10
11         [Benchmark]
12         public void TestC_Sharp_Sum() => MySum();
13
14         #region C_SharpFunctions
15         public static async void ReturnNumber()
16         {
17             var state = await CSharpScript.RunAsync("return 42;");
18             Console.WriteLine(state.ReturnValue);
19         }
20         public static async void MySum()
21         {
22             var state = await CSharpScript.RunAsync("return 3 + 3;");
23             Console.WriteLine(state.ReturnValue);
24         }
25         #endregion
26     }
27 }
```

Bogus

Bogus haben wir in unserem Projekt für die Generierung von Fake Daten benutzt. Wir haben damit Schüler- und Lehrernamen erstellen lassen die wir in unserer Anwendung als Testdaten benutzt haben. Bogus funktioniert ausschließlich für .NET-Sprachen wie C#, F# oder VB.NET.

Es ist ganz unkompliziert zu verwenden. Wir haben in unserer Arbeit nur Namen generieren lassen, jedoch könnte man zu jedem Namen noch eine ganze Menge hinzufügen wie zum Beispiel Telefonnummern, E-Mail-Adressen, Wohnadressen oder auch die Herkunft.

Folgendes Codebeispiel zeigt die Generierung unserer Fake Daten für eine Schulklasse:

Listing 4: Bogus

```
List<Student> firstStudents = new List<Student>();

#region Create Fake Students for each Schoolclass
for (int i = 0; i < 10; i++)
{
    var studentFaker = new Faker<Student>()
        .RuleFor(x => x.Name, x => x.Person.FullName)
        .Generate();
    firstStudents.Add(studentFaker);
}
```

In der RuleFor() Methode kann man genau die Sachen angeben die man benötigt. In unserem Fall haben wir nur Vornamen und Nachnamen benötigt.

4 Umsetzung

Siehe tolle Daten in Tab. 1.

Siehe und staune in Abb. 1. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

	Regular Customers	**Random Customers**
Age	20-40	>60
Education	university	high school

Tabelle 1: Ein paar tabellarische Daten

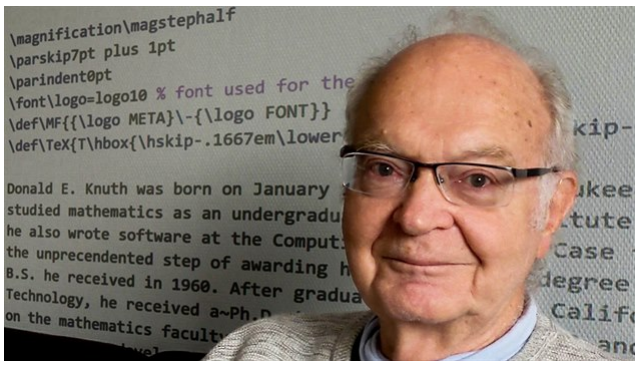

Abbildung 1: Don Knuth – CS Allfather

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus. Dann betrachte den Code in Listing 5.

Listing 5: Some code

```
# Program to find the sum of all numbers stored in a list (the not-Pythonic-way)

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

5 Zusammenfassung

Aufzählungen:

- Itemize Level 1
  - Itemize Level 2
    - Itemize Level 3 (vermeiden)

1. Enumerate Level 1
   a. Enumerate Level 2
      i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

Literaturverzeichnis

[1] P. Rechenberg, G. Pomberger *et al.*, *Informatik Handbuch*, 4. Aufl. München – Wien: Hanser Verlag, 2006.

[2] Association for Progressive Communications, „Wireless technology is irreplaceable for providing access in remote and scarcely populated regions," 2006, letzter Zugriff am 23.05.2021. Online verfügbar: http://www.apc.org/en/news/strategic/world/wireless-technology-irreplaceable-providing-access

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang