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:
Stefan Schwammal
Susi Schwammal

Betreuer:
Lukas Lehrer

Projektpartner:
Petra Programmierer, Initech Inc.

Leonding, April 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

2 Evaluierung von Skriptsprachen

2.1 Auswahl der Skriptsprachen

Die Wahl der Scriptsprachen wurde auf Grund einer intensiven Internet-Recherchen und Fachgesprächen mit Mitschüler*innen und Professor*innen gefällt.
Folgende Scriptsprachen wurden gewählt:

2.1.1 Lua

Lua ist eine extrem schnelle Programmiersprache, die für ihre hohe Ausführungsgeschwindigkeit geschätzt wird. Diese Schnelligkeit, kombiniert mit dem geringen Speicherbedarf der Sprache, macht sie ideal für ressourcenbeschränkte Umgebungen und eingebettete Systeme. Lua bietet eine "Out-of-the-Box"-Nutzbarkeit, die es Entwicklern ermöglicht, sofort nach der Installation loszulegen, ohne sich um eine Vielzahl von Abhängigkeiten kümmern zu müssen. Ihre Einbettbarkeit ist ein weiteres Kernelement, das sie besonders attraktiv für Softwareprojekte macht, die eine integrierte Skriptsprache benötigen. Besonders in der Spieleindustrie hat Lua sich als beliebte Wahl für das Scripting etabliert. Hier ermöglicht es Entwicklern, schnell interaktive und flexible Spielmechanismen zu implementieren, ohne die Hauptspiellogik zu beeinträchtigen. Als Open-Source-Software steht Lua zudem einer breiten Entwicklergemeinschaft zur Verfügung, die zur kontinuierlichen Verbesserung und Erweiterung der Sprache beiträgt. All diese Aspekte machen Lua zu einer vielseitigen und kraftvollen Option für eine Reihe von Anwendungen, insbesondere für das Scripting in Videospielen, wie in "World of Warcraft" oder "Roblox".

2.1.2 IronPython

IronPython ist eine Implementierung der Python-Programmiersprache, die auf dem .NET-Framework aufbaut, nicht auf der Java Virtual Machine (JVM). In Bezug auf Schnelligkeit kann IronPython, je nach Anwendungsfall, sowohl Vorteile als auch Nachteile gegenüber der standardmäßigen CPython-Implementierung haben. Da es auf dem .NET Framework basiert, kann es schneller sein, wenn es darum geht, mit anderen .NET-Anwendungen oder Bibliotheken zu interagieren. Beim Speicherverbrauch ist IronPython in der Regel nicht so effizient wie CPython, da das .NET-Framework mehr Overhead haben kann. Einer der größten Vorteile von IronPython ist seine Dynamik. Durch die Nutzung der Dynamic Language Runtime (DLR) von .NET kann IronPython dynamische Typen und späte Bindungen effizienter verwalten als einige andere Implementierungen. Dies ermöglicht eine enge Integration mit .NET-Bibliotheken und erleichtert die schnelle Entwicklung und Iteration von Code.

2.1.3 Csharpscript

CSharpScript ist eine Bibliothek, die die Ausführung von Csharp-Skripten ermöglicht, und stellt eine flexible Möglichkeit dar, Csharp-Code dynamisch auszuführen. Einer der großen Vorteile von CSharpScript ist, dass es sowohl in gehosteten als auch in eigenständigen Ausführungsmodellen eingesetzt werden kann. In einem gehosteten Modell kann das Skript innerhalb einer bestehenden Anwendung laufen und Objekte und Funktionen der Anwendung nutzen oder modifizieren. In einem eigenständigen Modell kann das Skript als unabhängige Anwendung ausgeführt werden. Ein weiteres wichtiges Merkmal ist die Kompatibilität mit .NET 5/Core und höheren Versionen. Dies ermöglicht eine bessere Leistung, erweiterte APIs und die Möglichkeit, plattformübergreifende Anwendungen zu entwickeln. Die Bibliothek stellt eine robuste Schnittstelle bereit, die die Integration von Csharp-Skripten in eine Vielzahl von Anwendungsdomänen vereinfacht.

2.1.4 Javascript

JavaScript ist eine äußerst populäre und vielseitige Programmiersprache, die vor allem in der Webentwicklung eingesetzt wird. Sie zeichnet sich durch ihre Einfachheit und Benutzerfreundlichkeit aus, was sie besonders für Einsteiger attraktiv macht. Die Sprache ist intuitiv aufgebaut, sodass die grundlegenden Konzepte schnell verstanden und angewendet werden können. Ein weiterer Vorteil ist, dass alle modernen Webbrowser eine JavaScript-Engine integriert haben. Dies ermöglicht es Entwicklern, Code unmittelbar im Browser auszuführen und zu testen, ohne zusätzliche Software installieren zu müssen. In Bezug auf die Sicherheit bietet JavaScript zwar einige Mechanismen, wie die Ausführung in einer Sandbox-Umgebung, um den Zugriff auf das Betriebssystem des Nutzers zu beschränken. Allerdings ist es wichtig, die Risiken von Cross-Site-Scripting (XSS) zu berücksichtigen, einer Art von Angriff, der durch schlecht geschützte JavaScript-Code ermöglicht wird. Daher ist es essentiell, bewährte Sicherheitspraktiken anzuwenden, um solche Schwachstellen zu minimieren. Insgesamt bietet JavaScript eine ausgewogene Mischung aus Benutzerfreundlichkeit, Intuitivität und Funktionalität, allerdings müssen Entwickler stets wachsam in Bezug auf Sicherheitsrisiken wie XSS sein.

2.2 Kriterienkatalog

Jede der untersuchten Sprachen hat ihre eigenen Vor- und Nachteile, und die Auswahl der besten Option kann je nach Projektanforderungen variieren. In diesem Abschnitt betrachten wir diese vier Sprachen im Kontext von .NET unter verschiedenen Aspekten:

- Aktivität der Entwicklung:
  - Wie lebendig und aktiv ist die Community hinter der Sprache?
  - Wie oft werden Updates veröffentlicht, und wie gut ist die Dokumentation?
- Funktionalität:
  - Welche Features bietet die Sprache, und wie reichhaltig ist ihr Ökosystem?
  - Gibt es umfangreiche Bibliotheken, die die Entwicklung erleichtern?
- Einsetzbarkeit:
  - Wie einfach lässt sich die Sprache in bestehende oder neue .NET-Projekte integrieren?
  - Welche Voraussetzungen müssen erfüllt sein, und wie komplex ist die Integration?
- Performance:
  - Wie steht es um die Laufzeit-Performance des Codes?
  - Inwiefern beeinflusst die Wahl der Sprache die Geschwindigkeit und Ressourceneffizienz der fertigen Anwendung?
- Debugging:
  - Welche Möglichkeiten bietet die Sprache für das Debugging von Code?
  - Wie effektiv lassen sich Fehler finden und beheben, und welche Werkzeuge stehen zur Verfügung?

Alle Daten wurden auf zwei unterschiedlichen Geräten gemessen.

Die Daten für IronPython und Lua wurden auf

PC A unter folgenden Voraussetzungen gemessen:

- Gerätspezifikationen:

Hersteller	HP
Gerätname	LAPTOP-5U1879KR
Prozessor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Installierter RAM	8.00 GB (7.89 GB verwendbar)
Produkt-ID	00325-81357-65742-AAOEM
Systemtyp	64-Bit-Betriebssystem, x64-basierter Prozessor

- Betriebssystem:

Edition	Windows 11 Home
Version	21H2
Installiert am	24.11.2021
Betriebssystembuild	220.001.098
Leistung	Windows Feature Experience Pack 1000.22000.1098.0

Die Daten für Csharpscript und Javascript wurden auf

PC B unter folgenden Voraussetzungen gemessen:

- Gerätspezifikationen:

Hersteller	Selber Gebaut
Gerätname	PC-Philipp
Prozessor	AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz
Installierter RAM	16.00 GB
Produkt-ID	00330-80000-00000-AA542
Systemtyp	64-Bit-Betriebssystem, x64-basierter Prozessor

- Betriebssystem:

Edition	Windows 10 Pro
Version	22H2
Installiert am	17.08.2020
Betriebssystembuild	19045.3393
Leistung	Windows Feature Experience Pack 1000.19044.1000.0

2.2.1 Aktivität der Entwicklung

In der nachfolgenden Tabelle wird dargestellt, wie intensiv die Entwicklerteams an den unterschiedlichen Scriptsprachen arbeiten und ob diese auch die Nuget-Pakete entwickeln.

Aktivität	IronPython	Lua	CsharpScripting	Javascript
Commits in den letzten 10 Monaten	179	27	702	1153
Nuget-Packages vom Sprachentwicklerteam selber	Nein	Nein	Ja	Nein
Releases in den letzten 10 Monaten	2 (2.7.1 und 3.4.0-beta1)	1 (v5.4.4)	v4.0.0 und höher	v4.6.2 (und höher)
Unterstützt aktuelle major Versionen von Scriptsprache	Ja	Ja	Ja	Ja
Unterstützt aktuelle Versionen von .NET	Ja	Ja	Ja	Ja

2.2.2 Einsetzbarkeit

Die folgende Tabelle stellt dar, auf welchen Betriebssystemen die veschiedenen Scriptsprachen mit .NET lauffähig sind.

Einsetzbarkeit	IronPython	Lua	CsharpScripting	Javascript
Auf Windows lauffähig	Ja	Ja	Ja	Ja
Auf MAC lauffähig	Ja	Ja	Ja	Ja
Auf Linux lauffähig	Ja	Ja	Ja	Ja

2.2.3 Performance

Der Speicherplatzt wurde aus dem Windows-File-Explorer entnommen. Die Geschwindigkeitsmessungen erfolgten mit Dotnet-Benchmark.

DotNetBenchmark ist nicht eine standardisierte Bibliothek oder ein offizielles Werkzeug, aber der Begriff könnte in der Kontext von .NET-Entwicklung für Benchmarking-Tests verwendet werden. In der Regel wird BenchmarkDotNet als die vorherrschende Bibliothek für das Benchmarking in der .NET-Umgebung angesehen. Es ermöglicht Entwicklern, die Leistung von .NET-Code einfach und genau zu messen. Mit BenchmarkDotNet können Microbenchmarks durchgeführt werden, die sehr spezifische Aspekte des Codes testen, wie zum Beispiel die Ausführungszeit einer Methode. Als eine einfache Anwendung wird ein Programm genommen, dass eine Funktion aufruft, welche 42 zurück gibt und dies anschließen auf die Konsole ausgibt.

Performance	IronPython	Lua	CsharpScripting	Javascript
Speicher einer einfachen Anwendung	13 MB	7.3 MB	168 KB	416 KB
Durchnittliche Laufzeit einer einfachen Anwendung	137.118 μs	8.088 μs	39.59 ms	128.14 ms
Durchnittliche Laufzeit einer Additionsfunktion	2340.688 μs	10.053 μs	39.59 ms	29.77 ms
Durchnittliche Laufzeit von Übergabe eines .NET-Objekts	9054.007 μs	7548.497 μs	66.72 ms	46.27

Es folgen nun die Resultate von Dotnet-Benchmark.

Für:

- Lua und IronPython

Method	Mean	Error	StdDev
TestIronPython	137.118 μs	7.5278 μs	21.959 μs
TestIronPythonSum	2,340.688 μs	71.9924 μs	208.863 μs
TestLua	8.088 μs	0.3702 μs	1.020 μs
TestLuaSum	10.053 μs	0.4560 μs	1.330 μs
TestLua-PassDotNetObject-AndCallFunction	7,548.497 μs	1,258.6385 μs	3,711.124 μs
TestIronPython-PassDotNetObject-AndCallFunction	9,054.007 μs	471.9551 μs	1,346.514 μs

–

- Csharpscript

Method	Mean	Error	StdDev
TestCSharpSimple	39.59 ms	0.354 ms	0.331 ms
TestCSharpSum	39.56 ms	0.321 ms	0.301 ms
TestCSharp- Objects	66.72 ms	0.875 ms	0.819 ms

–

- Javascript

Method	Mean	Error	StdDev
TestJavascriptSimple	128.14 ms	1,102.64 ms	286.35 ms
TestJavaScriptSum	29.77 ms	255.31 ms	66.30 ms
TestJavascript-DotNetObjects	46.27 ms	397.72 ms	103.29 ms

–

2.2.4 Funktionalität

In der nachfolgenden Tabelle sind die Recherche-Ergebnisse hinsichtlich der Funktionalität der Scriptsprachen in .NET dargestellt. Die Informationen wurden aus den offizellen Webseiten der Nuget-Pakete entnommen.

Funktion	IronPython	Lua	CsharpScripting	Javascript
Kann auf .NET Variablen zugreifen	Ja	Ja	Ja	Ja
Kann globale Variablen	Ja	Ja	Ja	Ja
Unterstützt Erweiterungspakete der Scriptsprache	Ja (nicht numpy und pandas!)	Ja	Ja	Ja

2.2.5 Debugging

In der nachfolgenden Tabelle ist dargestellt, welche Art des Debugging bei welcher Scriptsprache möglich ist.

Debugging	IronPython	Lua	CsharpScripting	Javascript
Debugging durch Ausgabe auf der Konsole	Ja	Ja	Ja	Ja
Debugging mit Break Points in Visual Studio	Ja	Nein	Nein	Nein
Debugging mit Break Points in Visual Studio Code	Ja	Nein	Nein	Nein

Es folgen nun Screenshots, um zu beweisen, dass die Angaben stimmen.

- Der Beweis für das Debugging mit NLua:

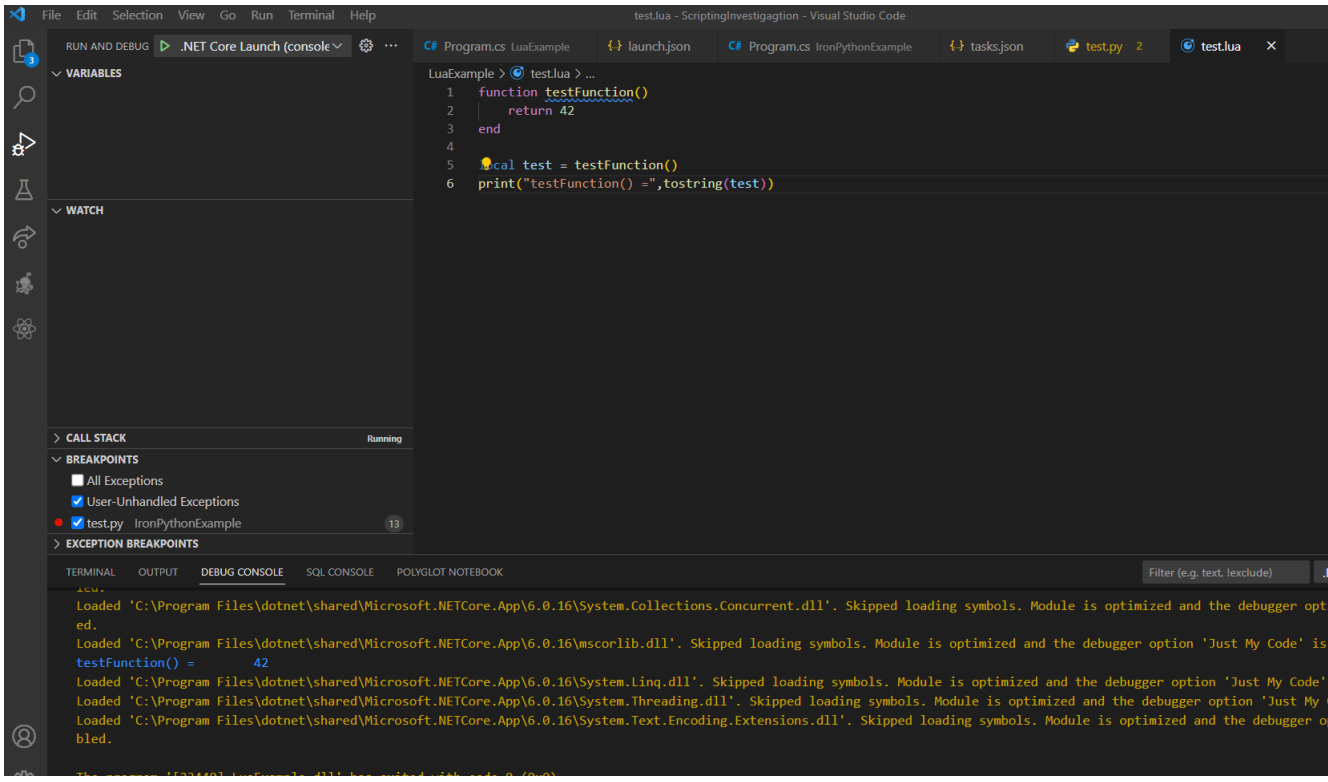

Abbildung 1: Lua-Konsolenausgabe

- Die Beweise für das Debugging mit IronPython:

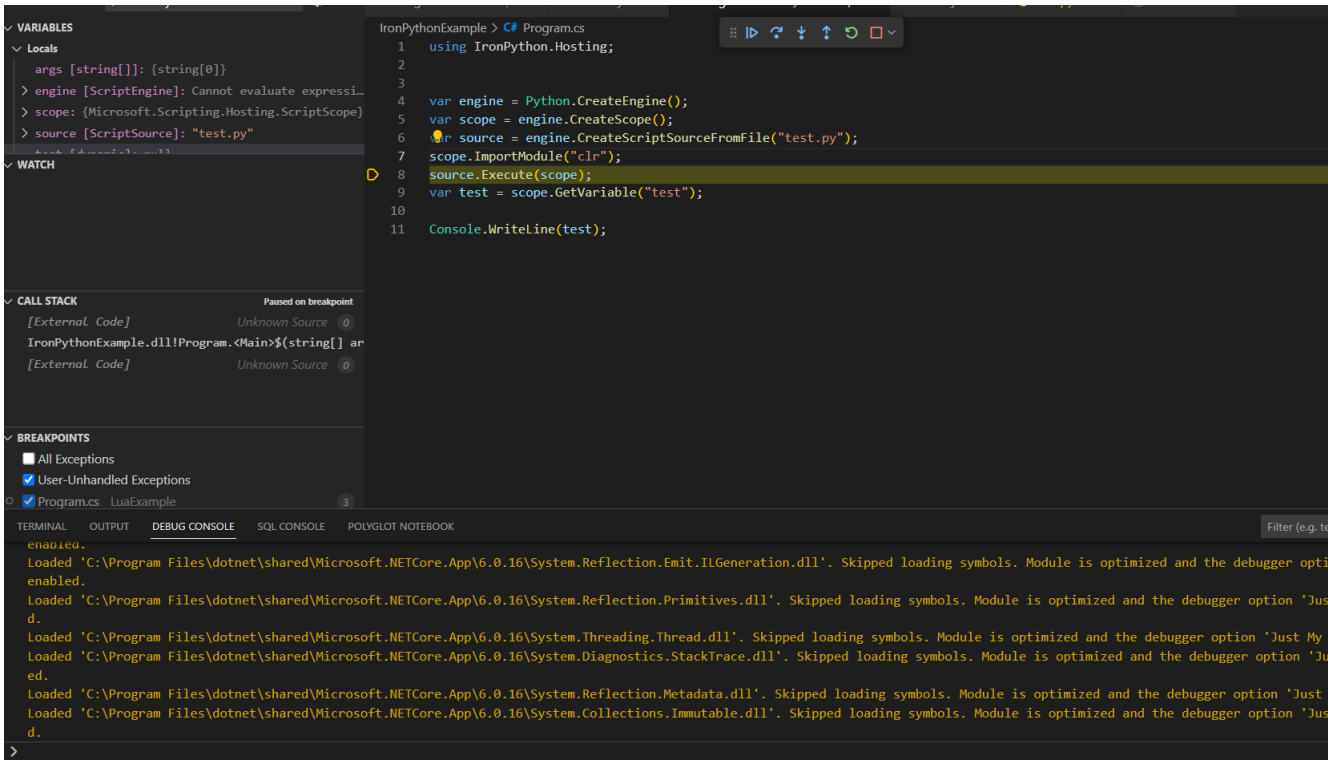

Abbildung 2: IronPython-VSCode-Breakpoint1

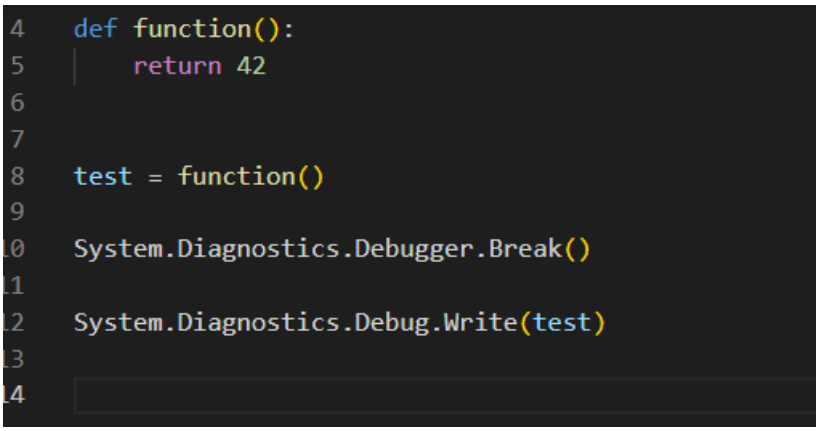

Abbildung 3: IronPython-VSCode-Breakpoint2

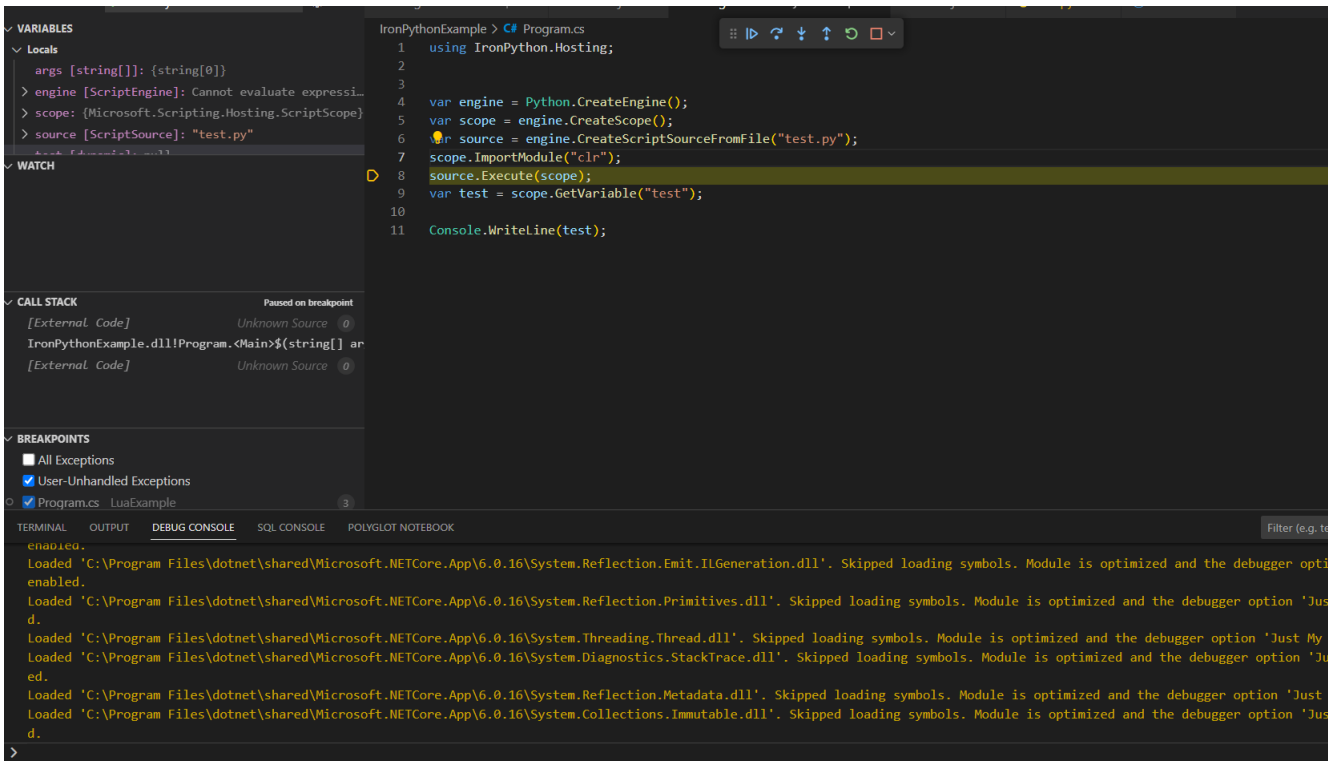

Abbildung 4: IronPython-VSCode-Breakpoint1

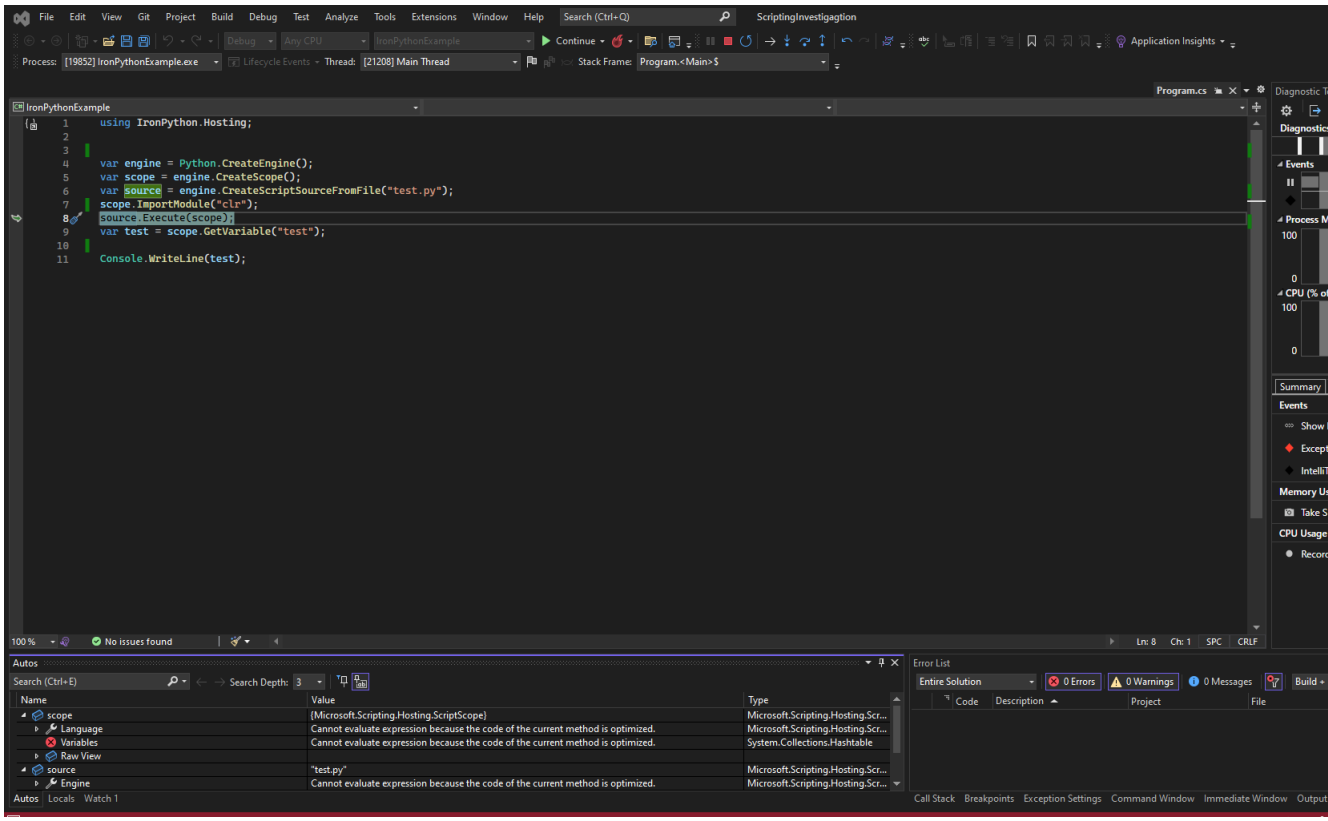

Abbildung 5: IronPython-VS-Breakpoint1

```
import clr
import System

def function():
    return 42


test = function()

System.Diagnostics.Debugger.Break()

System.Diagnostics.Debug.Write(test)
```

Abbildung 6: IronPython-VSBreakpoint2

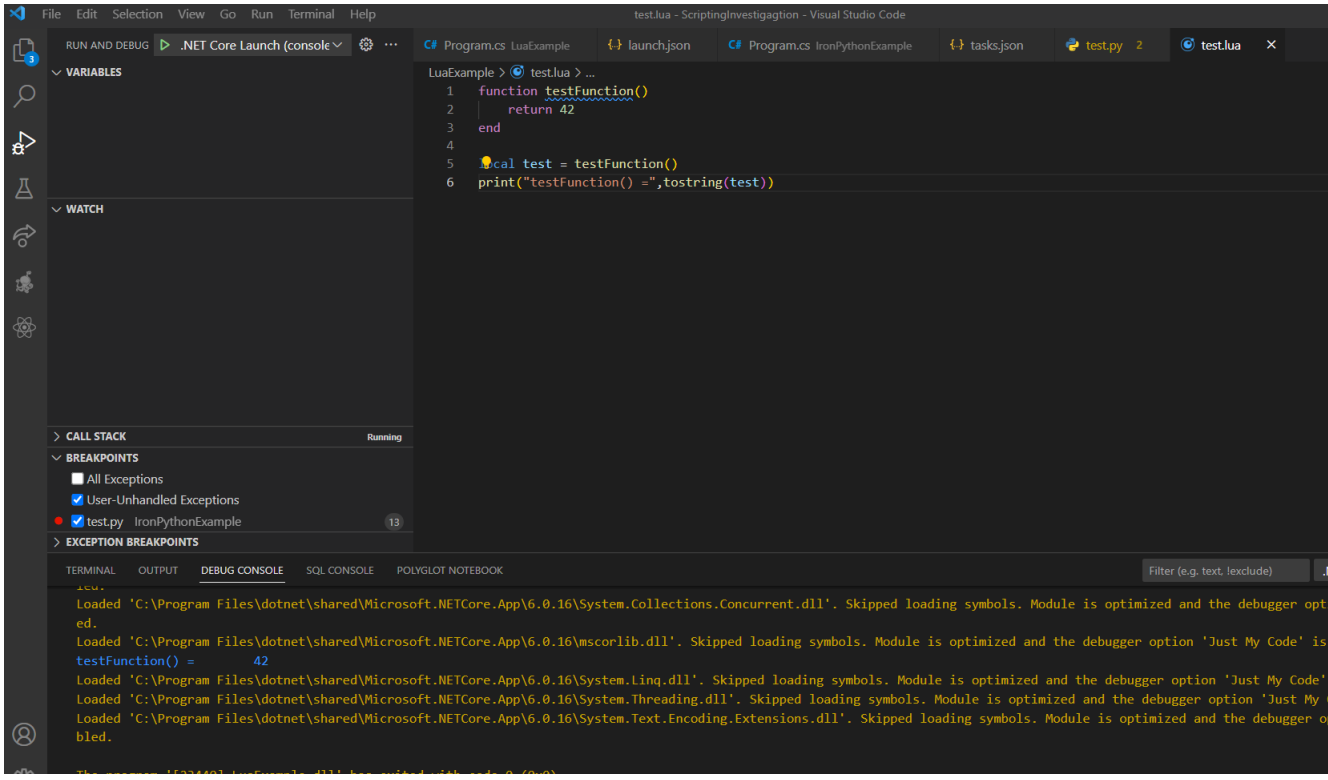

Abbildung 7: IronPython-Konsolenausgabe

3 Anwendung (Praxisteil)

3.1 Verwendete Technologien

Zur Realisierung der Beispielanwendung wurden folgende Technologien verwendet: Docker

- Docker
  - Docker ist eine Open-Source-Plattform, die es Entwicklern ermöglicht, Anwendungen in Containern zu erstellen und zu betreiben. Ein Container ist eine eigenständige, isolierte Einheit, die alles enthält, was eine Anwendung zur Ausführung benötigt, einschließlich Code, Laufzeitumgebung, Systembibliotheken und -tools. Docker vereinfacht die Bereitstellung und den Betrieb von Anwendungen, da es eine konsistente Umgebung bietet, die über verschiedene Systeme und Cloud-Dienste hinweg gleich bleibt.
- Docker-Compose
  - Docker Compose ist ein Tool für die Definition und Verwaltung von Multi-Container-Docker-Anwendungen. Es ermöglicht Entwicklern, eine docker-compose.yml Datei zu verwenden, in der sie die Dienste, Netzwerke und Volumes der Anwendung definieren können. Durch Ausführen eines einzigen docker-compose up Befehls können alle Dienste und Abhängigkeiten in der Reihenfolge gestartet werden, wie sie in der YAML-Datei definiert sind. Das Tool ist besonders nützlich für die Orchestrierung von Microservices und komplexen Anwendungen.

- ASP.NET
  - ASP.NET ist ein Web-Framework von Microsoft, das für die Entwicklung von Webseiten, Webanwendungen und Webdiensten verwendet wird. Es ist in der .NET-Plattform eingebettet und bietet eine Reihe von Bibliotheken und Tools für die schnelle und effiziente Entwicklung. ASP.NET kann mit verschiedenen Programmiersprachen wie Csharp, Fsharp und VB.NET verwendet werden. Es unterstützt sowohl MVC (Model-View-Controller) als auch Web API, was es zu einer vielseitigen Wahl für viele Arten von Webprojekten macht.
- AutoMapper
  - AutoMapper ist eine Open-Source-Bibliothek für die objekt-orientierte Programmierung, die automatische Zuordnungen zwischen zwei Objekten unterschiedlichen Typs ermöglicht. Es wird oft in Csharp-Projekten und im Kontext von .NET-Anwendungen verwendet. Durch die Verwendung von Konventionen statt der expliziten Konfiguration kann AutoMapper den Boilerplate-Code reduzieren, der normalerweise erforderlich ist, um Daten von einem Datenmodell in ein anderes zu übertragen. Es ist besonders nützlich in Szenarien wie dem Mapping zwischen Datenzugriffsobjekten (DAOs) und Data Transfer Objects (DTOs).

3.2 Aufbau

Um einen Überblick über die Beispielanwendung zu erhalten folgt nun ein Komponentendiagramm:

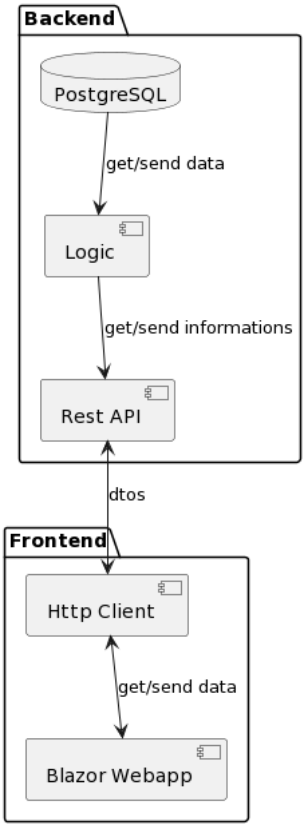

Abbildung 8: Komponenten – UML Diagramm

Damit der Aufbau der .NET-Solution noch klarer wird ist nun die YAML-Datei dargestellt:

```yaml
version: '3.8'
services:
  grades_db:
    image: postgres
    ports:
      - 5432:5432
    environment:
      POSTGRES_PASSWORD: postgres
      POSTGRES_USER: postgres
      POSTGRES_DB: GradeDb
    healthcheck:
          test: ["CMD-SHELL", "sh -c 'pg_isready -U postgres -d
              GradeDb'"]
          interval: 10s
          timeout: 3s
          retries: 55

  grades_backend:
    container_name: ${DOCKER_REGISTRY-}grades_backend
    image: grades_backend
    privileged: true
    ports:
      - 5000:80
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      -
          ConnectionStrings__DefaultConnection=UserID=postgres;Password=postgres;
          Security=true;Pooling=true;",

    build:
      context: .
      dockerfile: ./API/Dockerfile
    depends_on:
      grades_db:
        condition: service_healthy

  grades_web:
    image: ${DOCKER_REGISTRY-}grades_web
    container_name: grades_web
    volumes:
    - /Client:/Client
    ports:
      - 8080:80
    environment:
      - ASPNETCORE_ENVIRONMENT=Development

    build:
       context: .
       dockerfile: ./Client/Dockerfile
    depends_on:
      - grades_backend
```

Die verwendeten Entitäten und ihre Relationen im Backend sind in der folgenden Abbildung dargestellt.

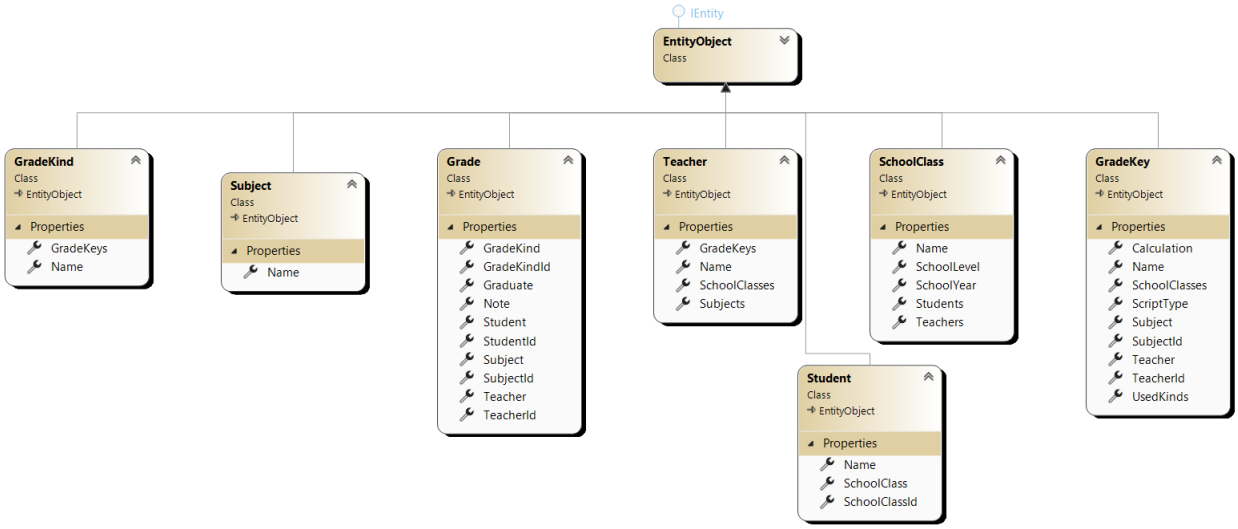

Abbildung 9: Entitäten – UML Diagramm

Es gibt viele Möglichkeiten Skripte in eine Anwendung zu importieren. Eine Möglichkeit ist die Skripte als Datei zu importieren. Anstatt alle benötigten Daten direkt in den Code einzubetten oder sie manuell über die Befehlszeile einzugeben, können Entwickler eine oder mehrere Dateien als Input verwenden, die das Skript dann liest und verarbeitet.

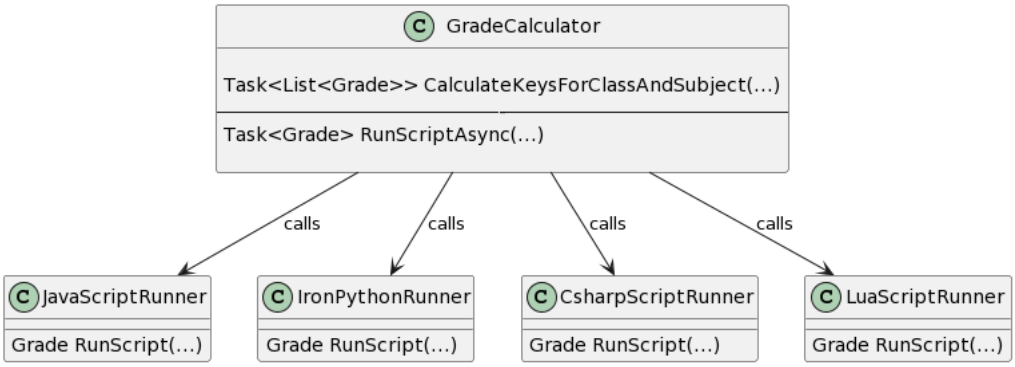

Abbildung 10: Logic Overview

Im folgenden Abschnitt werden einige Code-Ausschnitte betrachtet, die zeigen, wie man solche Datei-Übergaben in den untersuchten Scriptsprachen realisieren kann.

Listing 1: Code for Javascript

```
1   public class JavascriptRunner
2   {
3       public Grade RunScript(GradeKey key, List<Grade> grades)
4       {
5           var engine = new JintJsEngine();
6           Grade result = new Grade();
7           List<string>? logs = new List<string>();
8
9           try
10          {
11              // Definiere eine Variable im JavaScript-Code, um die
                    console.log-Ausgaben zu speichern
12              engine.Execute("var consoleOutput = [];");
13
14              // Definiere die console.log-Funktion im JavaScript-Code
15              engine.Execute(@"
16                              var console = {
17                                  log: function() {
18                                      consoleOutput.push(Array.from(arguments).join('
                                            '));
19                                  }
20                              };
21                          ");
22
23
24              var gradeKindsList = JsonConvert.SerializeObject(key.UsedKinds);
25              var gradesList = JsonConvert.SerializeObject(grades);
26
27              engine.SetVariableValue("gradeKindsList", gradeKindsList);
28              engine.SetVariableValue("gradesList", gradesList);
29
30              if (key.Calculation != null)
31              {
32                  engine.Execute(key.Calculation);
33              }
34
35              //Die Ausgabe der console.log-Anweisungen als JSON-String
36              string jsonOutput =
                    engine.Evaluate<string>("JSON.stringify(consoleOutput)");
37
38              // Konvertiere den JSON-String in eine Liste von strings
39              if (jsonOutput != null)
40              {
41                  logs = JsonConvert.DeserializeObject<List<string>>(jsonOutput);
42
43                  if (logs != null)
44                  {
45                      DisplayOutput(logs);
46                  }
47              }
48
49              // Get Return from Script
50              var resultGrade = engine.GetVariableValue("result");
51
52              result.Teacher = key.Teacher;
53              result.Graduate = Convert.ToInt32(resultGrade);
54          }
55          catch (Exception)
56          {
57              result.Teacher = null;
58              result.Graduate = 0;
59          }
60          return result;
61      }
62
63      private static void DisplayOutput(List<string> logs)
64      {
65          foreach (string output in logs)
66          {
67              Debug.WriteLine(output);
68          }
69      }
70  }
```

Listing 2: Code for NLua

```
1       /// <summary>
2       /// Runs lua-scripts
3       /// </summary>
4       public class LuaScriptRunner
5       {
6           private readonly Lua state;
7
8           public LuaScriptRunner()
9           {
10              this.state = new Lua();
11          }
12
13          public Grade RunScript(GradeKey key, List<Grade> grades)
14          {
15              if (key.Calculation == string.Empty || key.UsedKinds == null || grades
                    == null)
16              {
17                  throw new NullReferenceException("Not enough information for
                        Calculation");
18              }
19
20              var code = key.Calculation;
21
22              var result = new Grade();
23              try
24              {
25                  state.DoString(code);
26                  state.LoadCLRPackage();
27                  state["grades"] = grades;
28                  state.DoString(@"graduate = calculate()");
29                  result.Teacher = key.Teacher;
30                  var gr = state["graduate"];
31                  if (gr != null)
32                  {
33                      result.Graduate = Convert.ToInt32(gr);
34                  }
35              }
36              catch (Exception)
37              {
38                  throw;
39              }
40
41              return result;
42          }
43      }
```

Listing 3: Code for CsharpScripting

```
1       public class CsScriptRunner
2       {
3           public static Grade RunScript(GradeKey key, List<Grade> grades)
4           {
5               var result = new Grade();
6
7               // StringWriter erstellen, um die Ausgabe des Skripts zu erfassen
8               StringWriter sw = new StringWriter();
9
10              // Console.Out umleiten
11              TextWriter originalOut = Console.Out;
12              Console.SetOut(sw);
13              try
14              {
15                  dynamic script = CSScript.Evaluator
16                      .ReferenceAssemblyOf(typeof(GradeKey))
17                      .ReferenceAssemblyOf(typeof(Grade))
18                      .CompileCode(key.Calculation)
19                      .CreateObject("*");
20
21                  var res = script.Calculate(key, grades);
22                  result.Teacher = key.Teacher;
23                  result.Graduate = Convert.ToInt32(res);
24
25              }
26              catch (Exception)
27              {
28                  result.Teacher = null;
29                  result.Graduate = 0;
30              }
31              finally
32              {
33                  // Output ausgeben
34                  Debug.WriteLine(sw);
35              }
36
37              return result;
38          }
39      }
```

4 Technologien

4.1 Foo

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

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.

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.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum

placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

4.2 Bar

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis

viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

4.2.1 Deeper

Nicht mehr im Inhaltsverzeichnis.

Deepest

Vermeide mich.

5 Umsetzung

Siehe tolle Daten in Tab. 1.

Siehe und staune in Abb. 11. 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

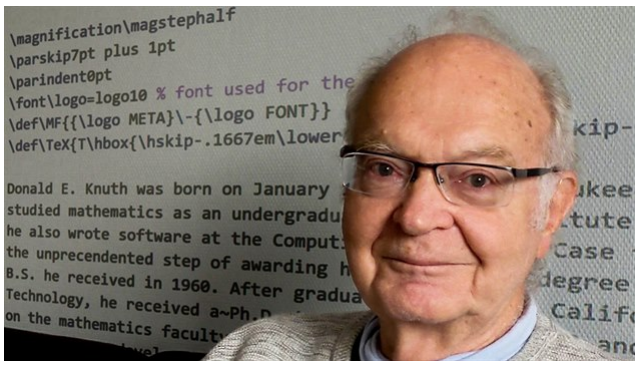

Abbildung 11: Don Knuth – CS Allfather

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 **??**.

6 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

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang