Anwendungsbeispiele

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Softwareentwicklung`
Teil:	`27/27`
Semester	`Sommersemester 2025`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	Zusammenfassung und Ausblick
Link auf den GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_Softwareentwicklung/blob/master/27_Anwendungen.md
Autoren	Sebastian Zug, Galina Rudolf & André Dietrich

Nachfrage: Secrets

Wie gehen wir mit Schlüsseln, Passwörtern usw. in unseren Codes um?

Zielstellung: + Komfortable Handhabung im Projekt + Projektübergreifende Verwendung (?) + Speicherung ohne Weiterleitung an Repositories

Ein Lösungsansatz ist die Verwendung von Microsoft.Extensions.Configuration.UserSecrets

```
dotnet new console -o secret_example
dotnet add package Microsoft.Extensions.Configuration.UserSecrets
dotnet user-secrets init
dotnet user-secrets set "ServiceAPIKey" "1213234435"
```

Das war es schon. Nun finden Sie unter

- `~/.microsoft/usersecrets/<user_secrets_id>/secrets.json` (Linux/macOS)
- `%APPDATA%\Microsoft\UserSecrets\<user_secrets_id>\secrets.json` (Windows)

den Eintrag

```
{
  "ServiceAPIKey": "1213234435"
}
```

Aus dem Programm heraus können Sie darauf unmittelbar zurückgreifen.

```
using Microsoft.Extensions.Configuration;

var config = new ConfigurationBuilder().AddUserSecrets<Program>().Build();
string APIsecret = config["ServiceAPIKey"];

Console.WriteLine(APIsecret);
```

Anwendungsbeispiel

Lassen Sie die Inhalte der Lehrveranstaltung anhand eines Codereviews Revue passieren lassen.

> Sie erhalten ein C# Programm und sollen es überarbeiten - welche Mängel finden Sie?

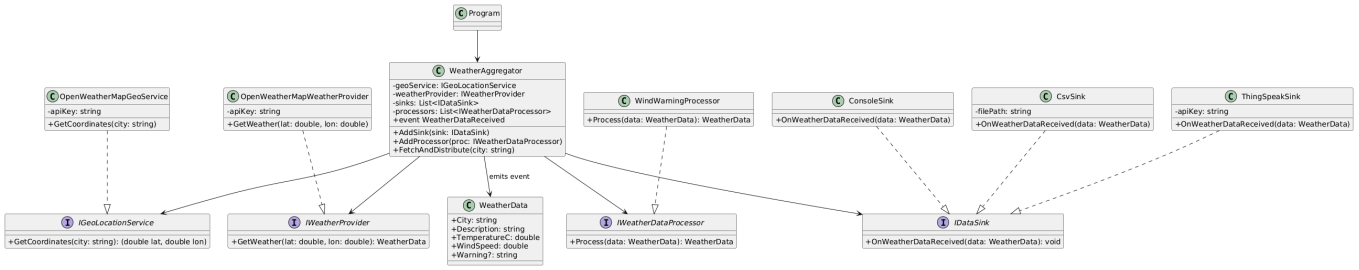
```
using System;
using System.Net;
using System.Text.Json;
using System.IO;

using Microsoft.Extensions.Configuration;

class Program
{
    static void Main()
    {
        var config = new ConfigurationBuilder()
            .AddUserSecrets<Program>()
```

```
        .Build();

    var apiKey = config["OpenWeatherMap:ApiKey"];
    var thingSpeakApiKey = config["ThingSpeak:ApiKey"]; // ThingSpeak W
      API Key
    string city = "Freiberg";

    WebClient client = new WebClient();
    string city_url = $"http://api.openweathermap.org/geo/1.0/direct?q
      ={city}&limit=1&appid={apiKey}";
    string city_json = client.DownloadString(city_url);
    Console.WriteLine(city_json);

    JsonDocument cityDoc = JsonDocument.Parse(city_json);
    JsonElement cityRoot = cityDoc.RootElement;
    double lat = cityRoot[0].GetProperty("lat").GetDouble();
    double lon = cityRoot[0].GetProperty("lon").GetDouble();

    string weather_url = $"http://api.openweathermap.org/data/2.5/weath
      ={lat}&lon={lon}&appid={apiKey}&units=metric";
    Console.WriteLine(weather_url);

    string weather_json = client.DownloadString(weather_url);
    Console.WriteLine(weather_json);

    JsonDocument weatherDoc = JsonDocument.Parse(weather_json);
    JsonElement weatherRoot = weatherDoc.RootElement;
    string wetter = weatherRoot.GetProperty("weather")[0].GetProperty
      ("description").GetString();
    double temp = weatherRoot.GetProperty("main").GetProperty("temp"
      ).GetDouble();
    double wind = weatherRoot.GetProperty("wind").GetProperty("speed"
      ).GetDouble();

    Console.WriteLine($"\nWetter in {city}:");
    Console.WriteLine($"Beschreibung: {wetter}");
    Console.WriteLine($"Temperatur: {temp}°C");
    Console.WriteLine($"Windgeschwindigkeit: {wind} m/s");

    // CSV-Datei erstellen/erweitern
    string csvFilePath = "wetterdaten.csv";
    string csvLine = $"{DateTime.Now:yyyy-MM-dd HH:mm:ss},{city},{wette
      ,{temp},{wind}";

    // Prüfen ob CSV-Datei bereits existiert
    if (!File.Exists(csvFilePath))
    {
        // Header schreiben falls Datei nicht existiert
        string header = "Datum,Stadt,Beschreibung,Temperatur_C
          ,Windgeschwindigkeit_ms";
        File.WriteAllText(csvFilePath, header + Environment.NewLine);
```

```
            File.WriteAllText(csvFilePath, header + Environment.NewLine);
        }

        // Wetterdaten anhängen
        File.AppendAllText(csvFilePath, csvLine + Environment.NewLine);

        Console.WriteLine($"\nWetterdaten wurden in '{csvFilePath}' gespeic
          );

        // ThingSpeak Daten senden
        if (!string.IsNullOrEmpty(thingSpeakApiKey))
        {
            string thingSpeakUrl = $"https://api.thingspeak.com/update?api_
              ={thingSpeakApiKey}&field1={temp}&field2={wind}";

            try
            {
                string thingSpeakResponse = client.DownloadString(thingSpea
                Console.WriteLine($"ThingSpeak Response: {thingSpeakRespons

                if (int.TryParse(thingSpeakResponse, out int entryId) && en
                  > 0)
                {
                    Console.WriteLine($"Daten erfolgreich an ThingSpeak ges
                      Entry ID: {entryId}");
                }
                else
                {
                    Console.WriteLine("Fehler beim Senden an ThingSpeak.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Fehler beim Senden an ThingSpeak: {ex
                  .Message}");
            }
        }
        else
        {
            Console.WriteLine("ThingSpeak API Key nicht gefunden. Daten wer
              nicht gesendet.");
        }
    }
}
```

SOLID

Erster Entwurf mit LLM


Program
WeatherAggregator
-geoService: IGeoLocationService
-weatherProvider: IWeatherProvider
-sinks: List<IDataSink>
-processors: List<IWeatherDataProcessor>
+event WeatherDataReceived
+AddSink(sink: IDataSink)
+AddProcessor(proc: IWeatherDataProcessor)
+FetchAndDistribute(city: string)
OpenWeatherMapGeoService
-apiKey: string
+GetCoordinates(city: string)
OpenWeatherMapWeatherProvider
-apiKey: string
+GetWeather(lat: double, lon: double)
WindWarningProcessor
+Process(data: WeatherData): WeatherData
ConsoleSink
+OnWeatherDataReceived(data: WeatherData)
CsvSink
-filePath: string
+OnWeatherDataReceived(data: WeatherData)
ThingSpeakSink
-apiKey: string
+OnWeatherDataReceived(data: WeatherData)
emits event
IGeoLocationService
+GetCoordinates(city: string): (double lat, double lon)
IWeatherProvider
+GetWeather(lat: double, lon: double): WeatherData
WeatherData
+City: string
+Description: string
+TemperatureC: double
+WindSpeed: double
+Warning?: string
IWeatherDataProcessor
+Process(data: WeatherData): WeatherData
IDataSink
+OnWeatherDataReceived(data: WeatherData): void

Resumee

Woche	Tag	SWE
1	4. April	Organisation, Einführung
2	7. April	Softwareentwicklung als Prozess
	11. April	Konzepte von Dotnet und C#
3	14. April	Elemente der Sprache C# I
	18. April	*Karfreitag*
4	21. April	*Ostermontag*
	25. April	Elemente der Sprache C# II
5	28. April	Strukturen / Konzepte der OOP
	2. Mai	Säulen Objektorientierter Programmierung
6	5. Mai	Klassenelemente in C# / Vererbung
	9. Mai	Klassenelemente in C# / Interfaces
7	12. Mai	Versionsmanagement im SWE-Prozess I
	16. Mai	Versionsmanagement im SWE_Pprozess II
8	19. Mai	Generics
	23. Mai	Container
9	26. Mai	UML Konzepte
	30. Mai	UML Diagrammtypen
10	2. Juni	UML Anwendungsbeispiel
	6. Juni	Testen
11	9. Juni	*Pfingstmontag*
	13. Juni	Dokumentation und Build Toolchains

12	16. Juni	Continuous Integration in GitHub
	20. Juni	Delegaten
13	23. Juni	Events
	27. Juni	Threadkonzepte in C#
14	30. Juni	Taskmodell
	4. Juli	Design Pattern
15	7. Juli	Language Integrated Query
	11. Juli	GUI - MAUI (leider nicht geschafft)

Aus die Maus

Danke für Ihr Interesse! Viel Erfolg bei den Prüfungen