

FH HAGENBERG

PROJEKTARBEIT

---

# Weather Tracer - Dokumentation

---

*Autor:*

Daniel ENGLISCH

Andreas ROITHER

*Übungsleiter:*

Daniel SKLENITZKA

1. Dezember 2018

Ausbaustufe 2 - Version 2.0



# Inhaltsverzeichnis

<b>1</b>	<b>UI Sketches</b>	<b>2</b>
1.1	Simulator . . . . .	2
1.1.1	Stationen auswählen . . . . .	2
1.1.2	Preset erstellen . . . . .	3
1.1.3	Presets zu Stationen zuweisen . . . . .	4
1.1.4	Simulationsübersicht . . . . .	5
1.2	Cockpit . . . . .	6
<b>2</b>	<b>Datenbankdesign</b>	<b>7</b>
2.1	Beispieldaten Generierung . . . . .	8
2.1.1	Stationen, Adressen und Communities . . . . .	8
2.1.2	Messdaten . . . . .	8
2.1.3	Andere Daten . . . . .	8
<b>3</b>	<b>Visual Studio Projektmappe</b>	<b>9</b>
3.1	Common.Dal.Ado . . . . .	10
3.2	Wetr.Dal.Interface . . . . .	11
3.3	Wetr.Dal.Ado . . . . .	12
3.4	Wetr.Dal.Factory . . . . .	13
3.5	Wetr.Domain . . . . .	14
3.6	Wetr.Test.Dal . . . . .	15
3.7	Wetr.Generator . . . . .	16
<b>4</b>	<b>Installationsanleitung</b>	<b>17</b>
4.1	Benötigte Programme und Voraussetzungen . . . . .	17
4.2	Datenbank . . . . .	17

# 1 UI Sketches

## 1.1 Simulator

Die Navigation im Simulator ist in vier Schritte eingeteilt:

- Auswählen von zu simulierenden Stationen
- Erstellen von Generierungspresets
- Zuweisen dieser Presets zu den einzelnen Stationen
- Simulieren der Messdatengenerierung mit Live-Visualisierung

Die einzelnen Stages werden mithilfe eines Tab-Controlls umgesetzt und man kann jederzeit, wenn die Simulation gestoppt ist, Einstellungen vornehmen.

### 1.1.1 Stationen auswählen

Alle verfügbaren Stationen werden in der rechten Liste (siehe Abbildung 1) angezeigt. Es ist für beide Spalten möglich, den Inhalt mit der sich oberhalb befindlichen Filtereingabe zu filtern, wobei auf mehrere Eigenschaften einer Station (Name, Postleitzahl, Besitzer, etc.) gefiltert wird. Die Pfeile zwischen den Spalten dienen nur zu visueller Betonung, dass die Stationen per Mausklick von der einen Spalte in die andere verschoben werden können. Nachdem alle zu simulierenden Stationen ausgewählt wurden, kann man entweder durch Klicken des "Weiter"-Buttons oder durch Auswählen des nächsten Tabs zum nächsten Schritt gewechselt werden.

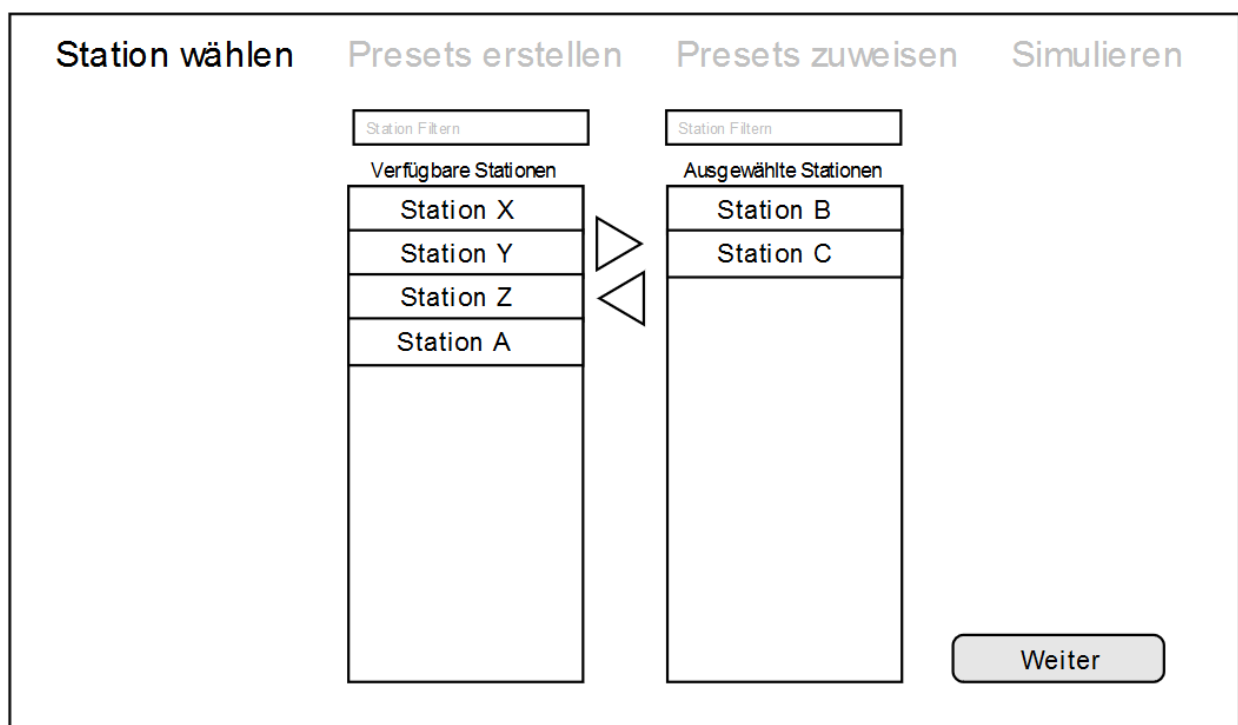


Abbildung 1: UI Sketch für das Auswählen von Stationen

### 1.1.2 Preset erstellen

Um ein Preset zu erstellen muss zunächst in die korrespondierenden Felder diverse Daten eingegeben werden. Für Minimalwert und Maximalwert bzw. Presetname wurden einfache Textfelder verwendet. Die Eingabe des Beginn- und Enddatums erfolgt durch ein Datetime Feld. Der Typ der zu generierende Messdaten, die Art der Verteilung und die Frequenz der Erstellung des Presets wird, wie in Abbildung 2 zu sehen, mit Dropdowns festgelegt. Wenn alle Presetdaten eingegeben wurde, kann das Preset mit dem "Hinzufügen"-Button in die darunter befindliche Tabelle eingefügt und auch wieder entfernt werden. Nach Abschluss dieses Schittes kann wie in der Vorherigen Ansicht auf den nächsten Schritt gewechselt werden.

Station wählen **Presets erstellen** Presets zuweisen Simulieren

Beginndatum

Enddatum

Minimalwert

Maximalwert

Messdatentyp ▼

Verteilung ▼

Frequenz ▼

Presetname

Hinzufügen

↓

Weiter

Presets

Preset A - (0-12°C) - Linear - Stündlich - 10.12.15 bis 15.1.16	X
Preset B - (1000-1600 hPa) - Zufall - Minütlich - 10.12.15 bis 15.1.16	X

Abbildung 2: UI Sketch für das Erstellen von Presets

### 1.1.3 Presets zu Stationen zuweisen

In diesem Schritt können den einzelnen Stationen beliebig viele Presets zugewiesen werden. In Abbildung 3 ist zu sehen, dass per Klick auf die Station in der linken Spalte sich der Text über der mittleren Spalte ändert, damit deutlich wird, dass nun Presets zur Station B zugewiesen werden können. Das Zuweisen bzw. Löschen von Presets für die ausgewählte Station funktioniert gleich wie im ersten Schritt beschrieben. Per Klicken auf einen Eintrag der rechten Spalte, wird dieser in die mittlere Spalte verschoben und umgekehrt. Die Pfeile dienen wieder zur Verdeutlichung der gewünschten Interaktion der beiden Spalten.

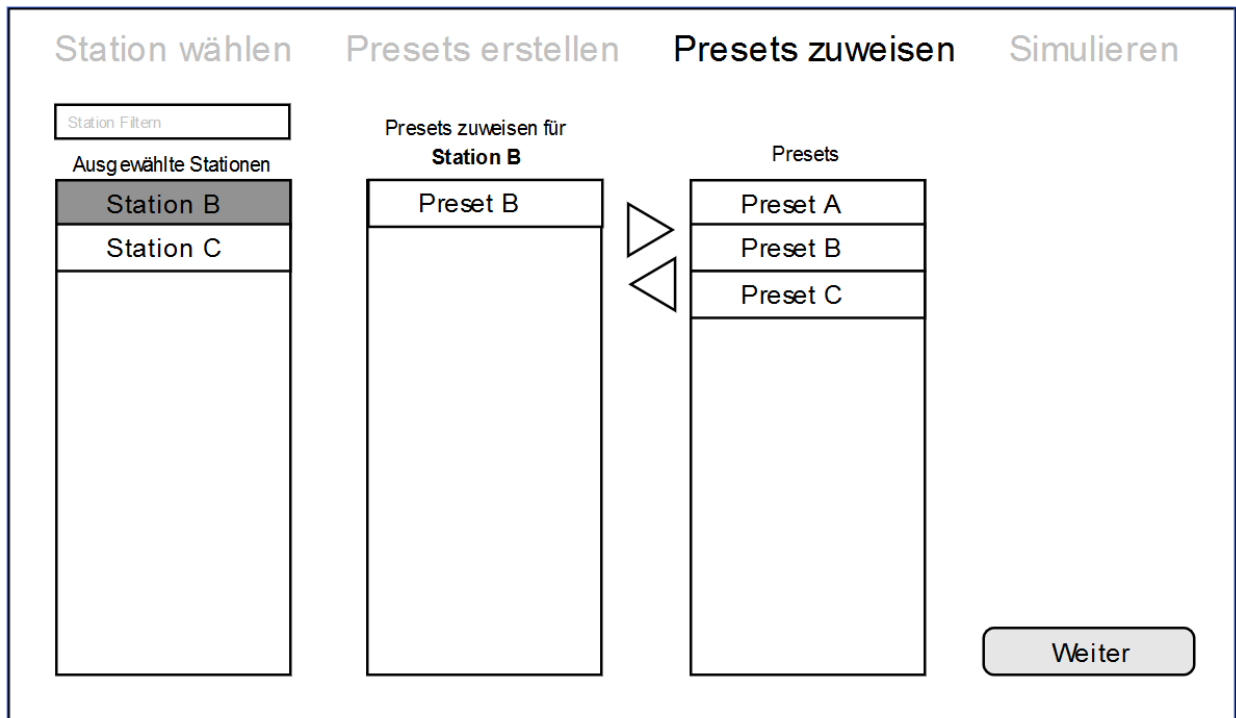


Abbildung 3: UI Sketch für das Zuweisen von Presets zu Stationen

### 1.1.4 Simulationsübersicht

Abbildung 4 zeigt die Liveansicht des Simulators in dem mit den zwei Buttons die Simulation gestartet oder gestoppt werden kann. Die Geschwindigkeit der Simulation, kann mit dem Slider zwischen Echtzeit und einem Vielfachen davon angepasst werden. Um bei hoher Last die Simulation zu entlasten, kann mit der Checkbox „Graphen berechnen“ die Live-Visualisierung deaktiviert werden. Ansonsten wird pro in der rechten Dropdownliste ausgewählten Station für jedes zugewiesene Preset ein Graph gezeichnet, der den Verlauf der generierten Messdaten darstellt. Rechts davon befindet sich eine Liste der zugewiesenen Presets, die deaktiviert werden können, um die Visualisierung übersichtlicher zu gestalten.

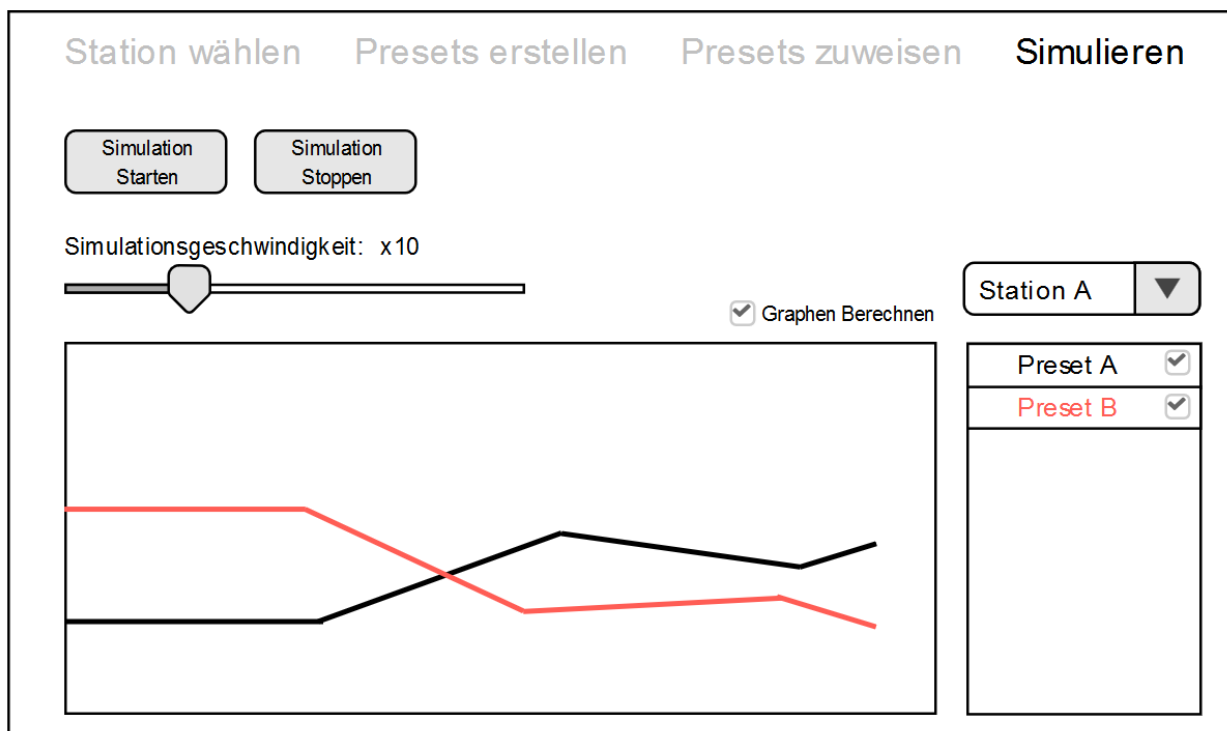


Abbildung 4: UI Sketch für die Simulationsübersicht

## **1.2 Cockpit**

## 2 Datenbankdesign

Dieses Projekt wurde mit einer MySQL Datenbank auf Version 5.7.23 realisiert. Es wurden zuerst die geforderten Entitäten aus der Angabe extrahiert und mithilfe eines grafischen Modellierungswerkzeugs namens MySQL Workbench <sup>81</sup> modelliert.

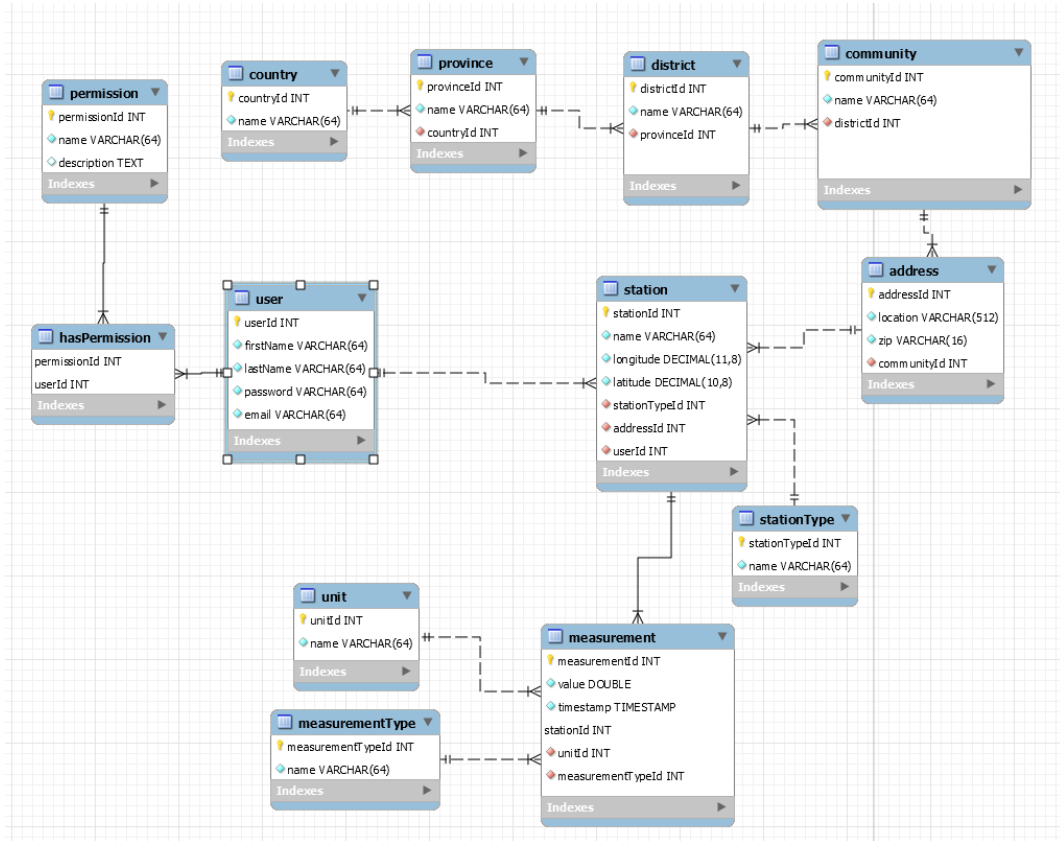


Abbildung 5: Datenbankschema des Wetr-Projekts in der ersten Ausbaustufe.

Wie in Abbildung 5 zu sehen wird zur Verwaltung des Standortes einer Station eine Reihe von abhängigen Entitäten verwendet. Um die Flexibilität zu erhöhen wurde neben zusätzlich ein *Country* modelliert. In einem *Country* befinden sich *Provinces*, welche Bundesländer darstellen. Jede *Province* wird in mehrere *Districts* unterteilt, ähnlich wie Bezirke. In jedem *District* gibt es mehrere *Communities*, welche mit Gemeinden vergleichbar sind. Als kleinste Entität in dieser Kette gibt es die *Address*, welche einen einfachen String zur Angabe von genaueren Adressdaten (rein zur Anzeige oder falls anderswo benötigt) und eine Zuordnung mittels Postleitzahl enthält.

Im Datenbankschema gibt es *User*, welche, falls benötigt, verschiedene *Permissions* zugewiesen haben können. Ein *User* kann mehrere *Stations* betreiben, welche wiederum neben der *Address* auch einen Namen und die Geokoordinaten in Form von Latitude und Longitude gespeichert hat. Der Typ der Station wurde in eine eigene Entität *StationType* ausgelagert.

Jede *Station* kann beliebig viele *Measurements* generieren, welche neben den ebenfalls ausgelagerten Entitäten *MeasurementType* und *Unit*, auch einen Zeitstempel und dazugehörigen Messwert besitzen.

<sup>81</sup><https://dev.mysql.com/downloads/workbench/>



## 2.1 Beispieldaten Generierung

### 2.1.1 Stationen, Adressen und Communities

Der *Extractor* (befindet sich im *extractor* Ordner) wurde mit Python<sup>2</sup> geschrieben und verwendet die Stationsliste der *Zentralanstalt für Meteorologie und Geodynamik*<sup>3</sup>. Die “.csv” Datei wird vom *Extractor* eingelesen und für jede *Station* wird eine Insert Anweisung in eine “.txt” Datei geschrieben. Zusätzlich wird anhand des Längen- und Breitengrades der Ort mit einem geolocator der Aufenthaltsort der Station ermittelt. Anhand dieser Daten werden SQL Anweisungen für die *Community* und *Address* Tabellen erstellt die von der jeweiligen Stationen referenziert werden.

### 2.1.2 Messdaten

Für die erste Ausbaustufe dieses Projekts wurde ein Generator implementiert, der über eine Millionen Messdaten generiert. Diese Messdaten sind jedoch nicht realitätsnahe, sondern haben als Basis den Jahresdurchschnitt in Österreich laut Klimatabelle<sup>4</sup>. Für eine genauere Beschreibung siehe Abschnitt 3.7.

### 2.1.3 Andere Daten

Die Beispieldaten der restlichen Tabellen wurden per Hand mithilfe vom Internet zusammengestellt. *Provinces*, *Districts* und weiter Standortbezogene Daten sind höchstwahrscheinlich nicht vollständig übernommen worden.

---

<sup>2</sup><https://www.python.org/>

<sup>3</sup><https://www.zamg.ac.at/cms/de/klima/messnetze/wetterstationen>

<sup>4</sup><https://www.klimatabelle.info/europa/oesterreich>

### 3 Visual Studio Projektmappe

Das gesamte Projekt befindet sich in einer Visual Studio Projektmappe, welche in folgende Unterprojekte gegliedert ist:

- **Common.Dal.Ado:** Hier befinden sich Hilfsklassen, die den Umgang mit ADO.NET leichter gestalten.
- **Wetr.Domain:** Dieses Projekt beinhaltet die Domainobjekte, welche als einfache Datenbehälter genutzt werden.
- **Wetr.Dal.Interface:** In diesem Paket befinden sich die Interfaces für die Datenzugriffsschicht für jedes Domainobjekt bzw. Tabelle.
- **Wetr.Dal.Ado:** Die ADO.NET Implementierung der Datenzugriffsschicht Interfaces.
- **Wetr.Dal.Factory** Beinhaltet Factories um die Instanziierung der benötigten konkreten Klassen zu abstrahieren.
- **Wetr.Test.Dal** Dieses Projekt beinhaltet Unit-Tests für die Datenzugriffsschicht.
- **Wetr.Generator:** Zuständig für das Generieren von Messdaten für die einzelnen Stationen.

### 3.1 Common.Dal.Ado

Das *Common.Dal.Ado* Projekt verwendet die Klassen *AdoTemplate*, *DefaultConnectionFactory*, *IConnectionFactory*, *Parameter* und *RowMapper*. In der *AdoTemplate* Klasse werden alle anderen Klassen verwendet. Mithilfe *IConnection* wird eine Verbindung zu Datenbank aufgebaut. *AdoTemplate* stellt eine Funktion *QueryAsync* bereit um eine Datenbankabfragen durchführen zu können.

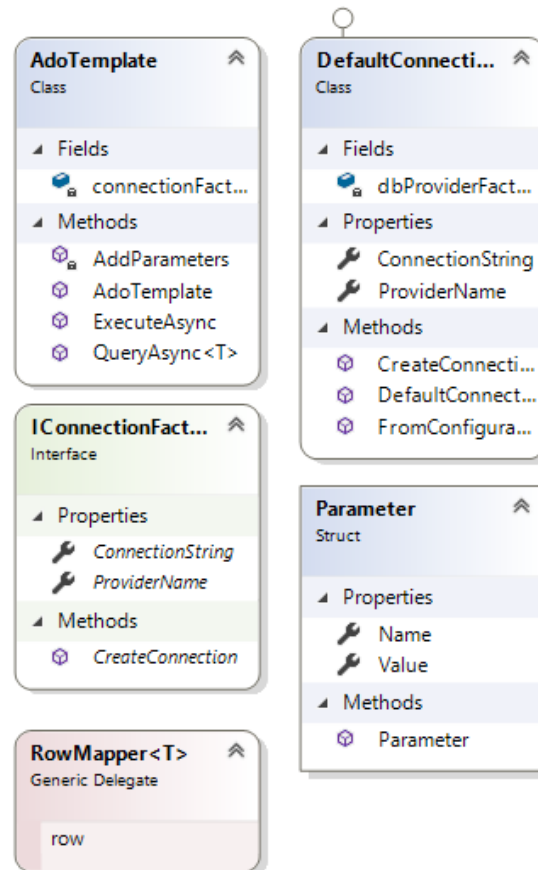


Abbildung 6: Common.Dal.Ado UML Diagramm

### 3.2 Wetr.Dal.Interface

Im *Wetr.Dal.Interface* Projekt wird für jede Klasse in *Wetr.Domain* ein eigenes Interface bereitgestellt das von den jeweiligen *Dao* Objekten implementiert wird. Jedes Interface implementiert *IDaoBase<T>*. Dieses Interface fasst die Methoden, die in jedem abgeleiteten *Dao Objekt* implementiert werden müssen, zusammen (FindAll, FindById, Delete, Insert, Update).

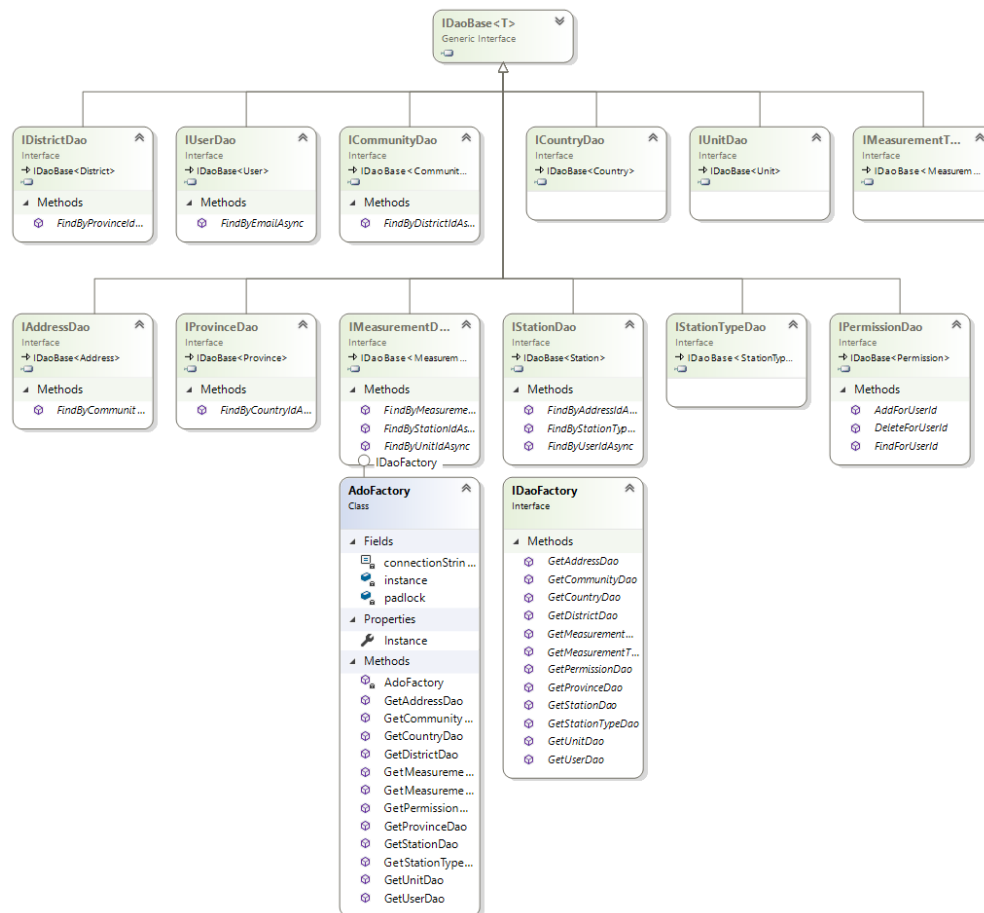


Abbildung 7: Wetr.Dal.Interface UML Diagramm

### 3.3 Wetr.Dal.Ado

Mit den Klassen im *Wetr.Dal.Ado* Projekt kann eine Verbindung zur Datenbank aufgebaut werden und spezielle SQL Operationen ausgeführt werden. Für jedes *Dao* Objekt gibt es eine *Wetr.Domänen* Klasse. Diese Klassen werden als Behälter Klassen für die angefragten Daten der Datenbank verwendet.

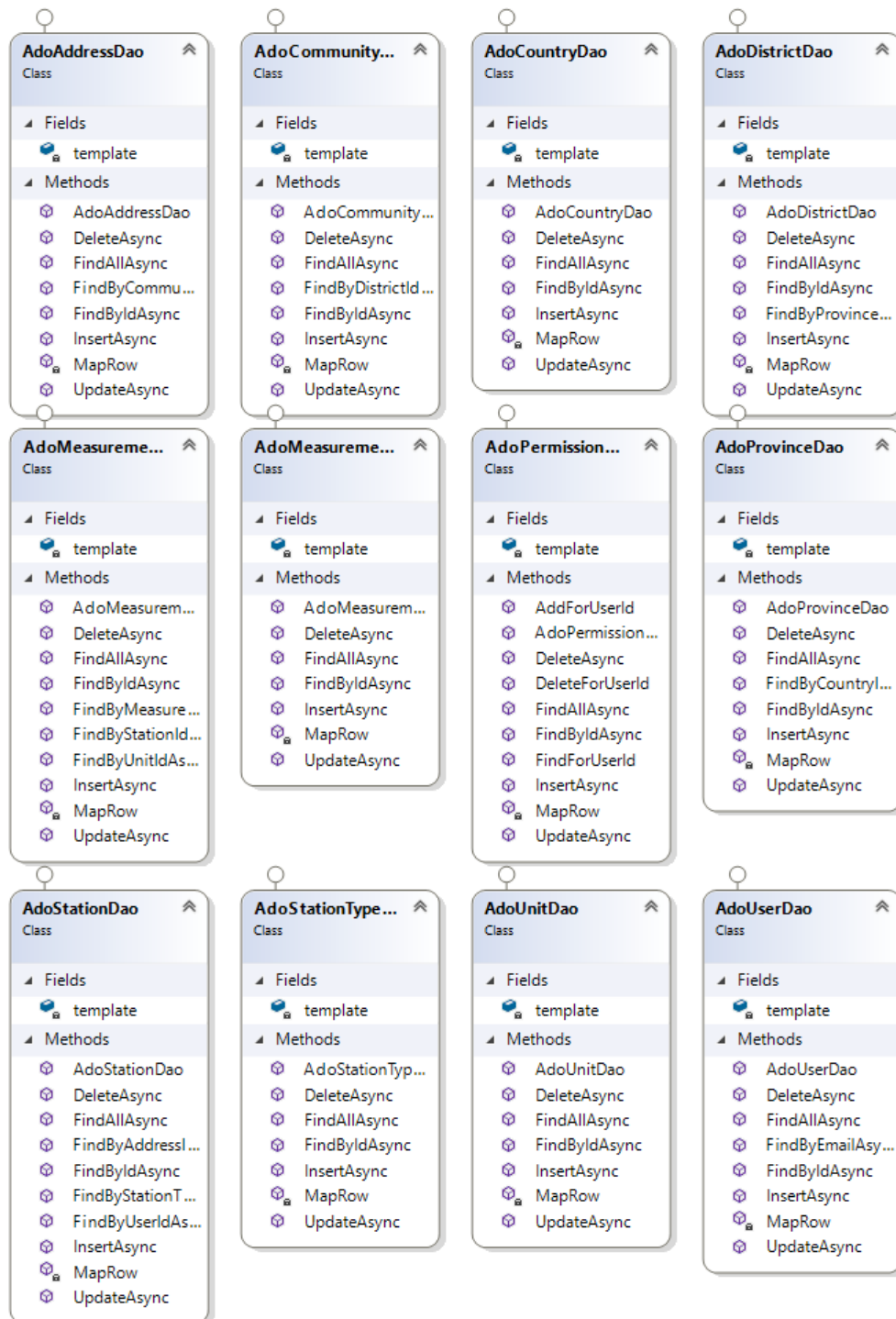


Abbildung 8: Wetr.Dal.Ado UML Diagramm

### 3.4 Wetr.Dal.Factory

Im *Wetr.Dal.Factory* Projekt wird eine Klasse *AdoFactory* bereitgestellt mit deren Hilfe ein *Dao* Objekt erstellt werden kann.

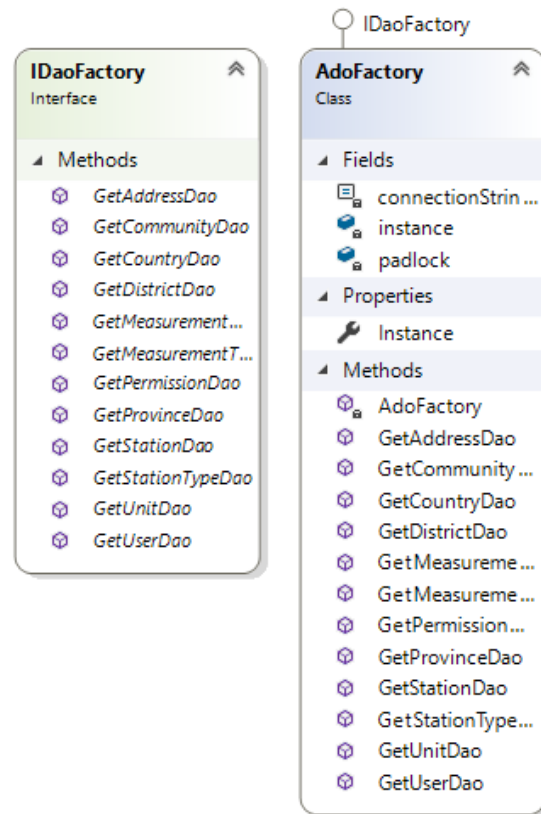


Abbildung 9: Wetr.Dal.Factory UML Diagramm

### 3.5 Wetr.Domain

Das *Wetr.Domain* Projekt enthält alle Behälterklassen für alle *Dao* Objekte.

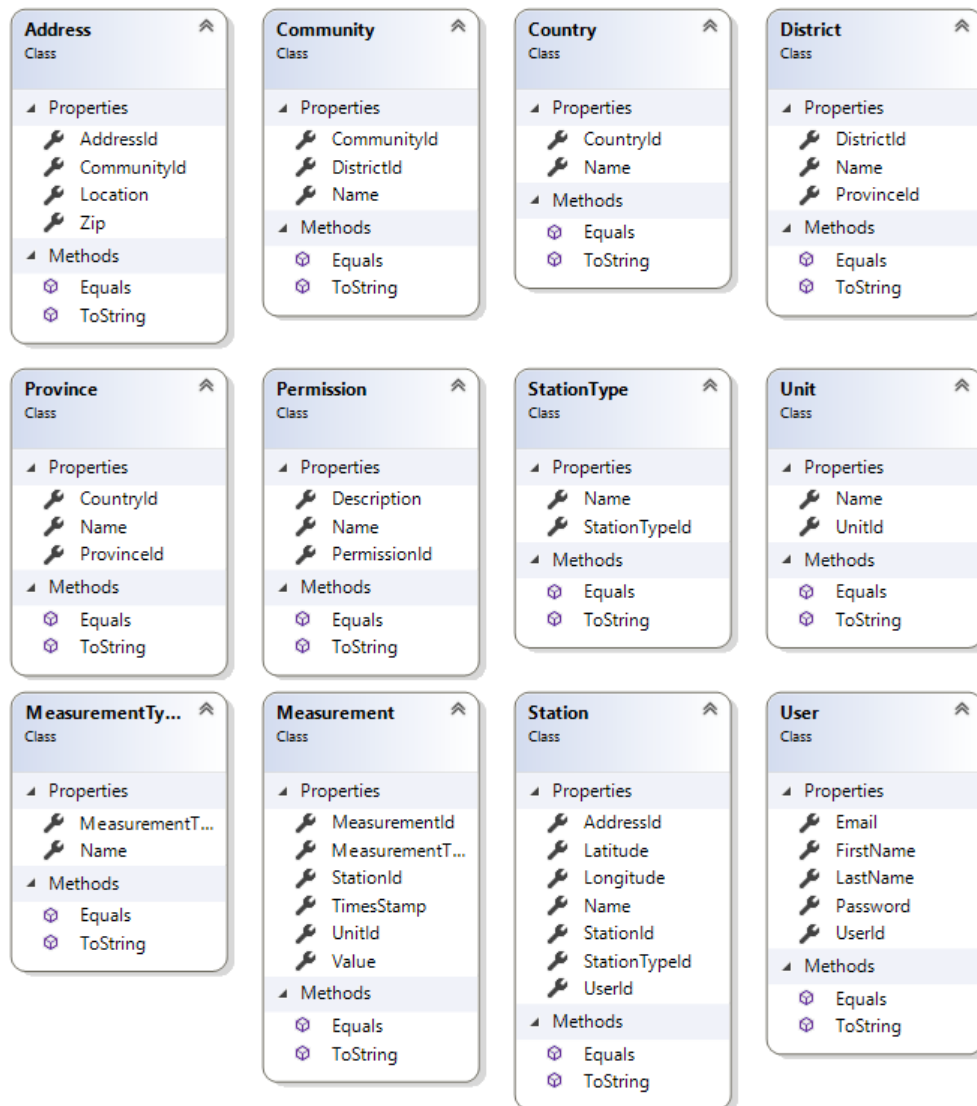


Abbildung 10: Wetr.Domain UML Diagramm

### 3.6 Wetr.Test.Dal

Beim *Wetr.Test.Dal* Projekt werden alle Funktionen von jedem *Dao* Objekt getestet. Alle Tests wurden von der Klassen *DaoBaseTest* abgeleitet, welche Test für *Dao*-Methoden forcieren, welche in allen Daos implementiert wurden.

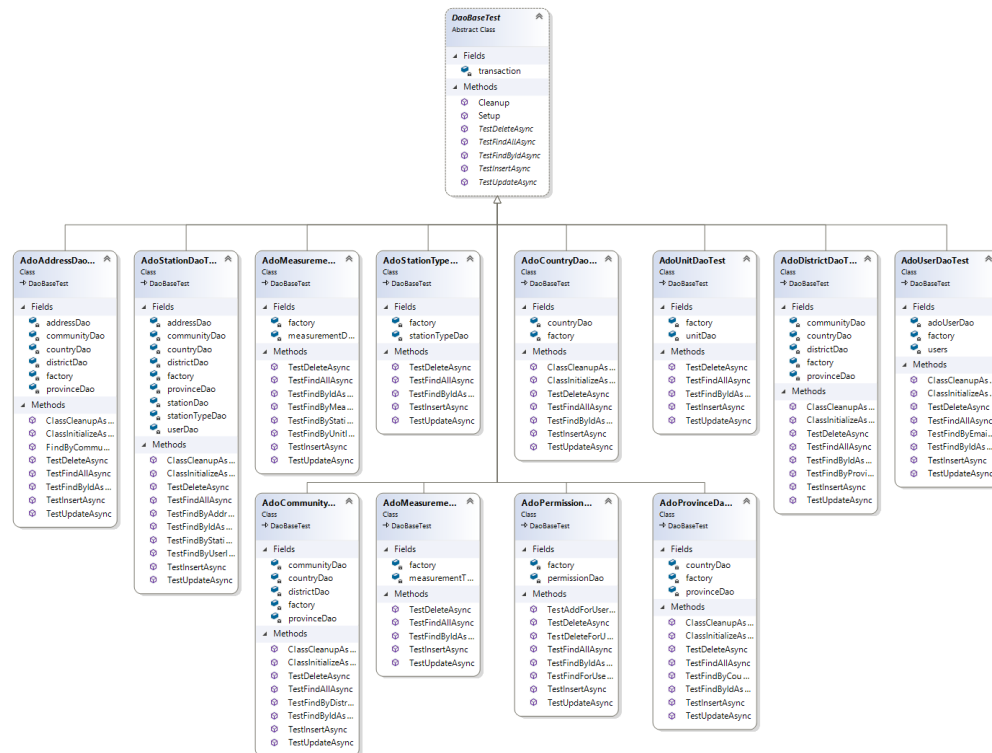


Abbildung 11: Wetr.Domain UML Diagramm

Wetr (74 Tests)		
✓	Wetr.Test.Dal (74)	628 ms
✓	Wetr.Test.Dal (74)	628 ms
✓	AdoAddressDaoTest (6)	50 ms
✓	AdoCommunityDaoTest (6)	46 ms
✓	AdoCountryDaoTest (5)	47 ms
✓	AdoDistrictDaoTest (6)	47 ms
✓	AdoMeasurementDaoTest (8)	51 ms
✓	AdoMeasurementTypeDao... (5)	44 ms
✓	AdoPermissionDaoTest (8)	109 ms
✓	AdoProvinceDaoTest (6)	46 ms
✓	AdoStationDaoTest (8)	55 ms
✓	AdoStationTypeDaoTest (5)	34 ms
✓	AdoUnitDaoTest (5)	50 ms
✓	AdoUserDaoTest (6)	45 ms

Abbildung 12: Beweis, dass die UnitTests erfolgreich durchlaufen.



### 3.7 Wetr.Generator

Der Generator generiert pro *MeasurementType* eine eigene Datei, in der sich die generierten Messdaten befinden. Pro Typ werden nicht genau gleich viele Messdaten generiert, somit ergibt sich eine Summe von etwa 1.2 Millionen Messdaten. Das Format der erzeugten Dateien ist so gestaltet, damit es mit einem speziellen SQL Befehl mittels Bulk-Insert<sup>5</sup> sehr schnell in die Datenbank aufgenommen werden kann. Die generierten Daten haben einen Zeitstempel, der sich innerhalb von einem Jahr bewegt.

#### Temperatur

Es wird jede Stunde ein Messwert generiert, der je nach Jahreszeit die Temperatur nach natürlichem Verlauf, sprich zur Mittagszeit ist es am wärmste und in der Nacht am kältesten, gestaltet. Die Werte beinhalten eine zufällige Abweichung von  $\pm 1.25$ . Der Minimal- bzw. Maximalwert wird pro Jahreszeit nach klimatabelle.info festgelegt.

#### Luftfeuchtigkeit

Die Berechnung der Luftfeuchtigkeit funktioniert gleich, wie die der Temperatur, nur, dass die durchschnittlichen Luftfeuchtigkeitswerte hergenommen wurden. Als zufällige Abweichung wurden  $\pm 10\%$  gewählt.

#### Niederschlag

Da nur der jährliche Durchschnittsniederschlag pro Jahreszeit zur Verfügung stand, wurden nicht stündlich, sondern täglich ein Messwert generiert. Dieser Wert bewegt sich zwischen 0 und der Anzahl des durchschnittlichen Tagesniederschlags Mal zwei.

#### Luftdruck

Jede Stunde wird ein Luftdruckwert generiert, der zufällig zwischen 900 und 1100 Hektopascal liegt.

#### Windrichtung

Die Windrichtung ändert sich in dieser einfachen Simulation jede Stunde und kann von 0 bis 360 Grad betragen.

#### Windstärke

Die Generierung der Windstärke ist in der aktuellen Ausbaustufe sehr primitiv gehalten und ändert sich stündlich zufällig von 0 und 20 *km/h*

---

<sup>5</sup><https://stackoverflow.com/questions/14330314/bulk-insert-in-mysql>

## 4 Installationsanleitung

### 4.1 Benötigte Programme und Voraussetzungen

Für dieses Projekt wird zusätzliche Software benötigt:

- Docker <sup>6</sup>
- Visual Studio <sup>7</sup>
- MySql-Connector (optional, nur falls notwendig) <sup>8</sup>

Um die Datenbank zu erstellen muss Docker gestartet sein. Die Datenbank kann mit dem *Powershell-Script* "run.ps1" automatisch generiert werden. Falls dieses Script wegen fehlender Berechtigungen nicht ausgeführt werden kann, muss eine *Shell* (Git-Bash oder ähnliches) im Ordner mit der Docker Compose Datei "docker-compose.yaml" geöffnet und folgenden Befehle nacheinander ausgeführt werden:

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker-compose up --build --force-recreate
```

### 4.2 Datenbank

Für dieses Projekt werden zwei Datenbanken benötigt, wobei die zweite eine Unit-Test Datenbank ist, die nur zur Ausführung solcher Tests benötigt wird.

Alle SQL Skripts müssen im *PhpMyAdmin* Interface unter dem *Import* Tab ausgeführt werden. *PhpMyAdmin* ist unter der Adresse "http://localhost:8080/" erreichbar.

Die benötigten Datenbanken können mit Hilfe von den zwei Skripten „create\_wetr.sql“ und „create\_wetr-unit-testing.sql“, welche sich im sql/Create Ordner befinden, erstellt werden.

Nach Ausführen der Create-Skripts können für die Produktivdatenbank (wetr) Beispieldaten eingefügt werden. Dazu die Datenbank auswählen und wieder in den Import Tab wechseln. Die zum Einfügen benötigte Datei „InsertEverythingWithoutMeasurement.sql“ befindet sich im sql/Insert Ordner und fügt alle Beispieldaten, außer die der Messwerte, in die Datenbank ein. Die Messdaten müssen zuerst generiert werden. Hierzu muss die Datei „Wetr.Generator.exe“ ausgeführt werden. Danach sollten sich sechs *.bulk* Dateien im Verzeichnis befinden, welche alle Messdaten für die verschiedenen Kategorien beinhalten. Um die Daten in die Datenbank einzufügen müssen die Befehle im Skript „InsertMeasurements.sql“ in *PhpMyAdmin* ausgeführt werden.

---

<sup>6</sup><https://www.docker.com/>

<sup>7</sup><https://visualstudio.microsoft.com/de/>

<sup>8</sup><https://dev.mysql.com/get/Downloads/Connector-Net/mysql-connector-net-8.0.13.msi>

```
LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsDownfall.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsHumidity.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsTemperature.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsWind.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsWindDirection.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsPreassure.bulk"
    INTO TABLE measurement
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';
```