

FH HAGENBERG

PROJEKTARBEIT

Weather Tracer - Dokumentation

Autor:

Daniel ENGLISCH
Andreas ROITHER

Übungsleiter:

Daniel SKLENITZKA

18. November 2018

Ausbaustufe 1 - Version 1.0



Inhaltsverzeichnis

1	Datenbankdesign	2
1.1	Beispieldaten Generierung	3
1.1.1	Stationen, Adressen und Communities	3
1.1.2	Messdaten	3
1.1.3	Andere Daten	3
2	Visual Studio Projektmappe	4
2.1	Common.Dal.Ado	5
2.2	Wetr.Dal.Interface	6
2.3	Wetr.Dal.Ado	7
2.4	Wetr.Dal.Factory	8
2.5	Wetr.Domain	9
2.6	Wetr.Test.Dal	10
2.7	Wetr.Generator	11
3	Installationsanleitung	12
3.1	Benötigte Programme und Voraussetzungen	12
3.2	Datenbank	12

1 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 ⁸ modelliert.

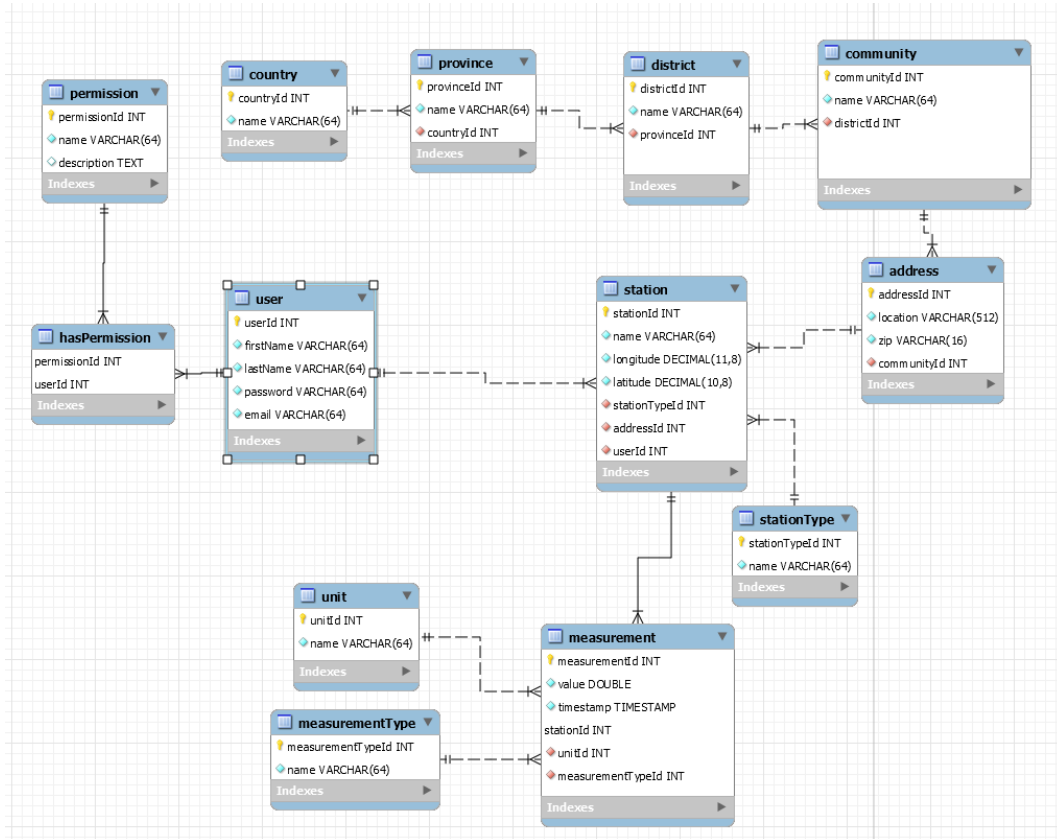


Abbildung 1: Datenbankschema des Wetr-Projekts in der ersten Ausbaustufe.

Wie in Abbildung 1 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.

⁸<https://dev.mysql.com/downloads/workbench/>

1.1 Beispieldaten Generierung

1.1.1 Stationen, Adressen und Communities

Der *Extractor* (befindet sich im *extractor* Ordner) wurde mit Python² geschrieben und verwendet die Stationsliste der *Zentralanstalt für Meteorologie und Geodynamik*³. 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.

1.1.2 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 Klimatabelle⁴. Für eine genauere Beschreibung siehe Abschnitt 2.7.

1.1.3 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.

²<https://www.python.org/>

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

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

2 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.

2.1 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.

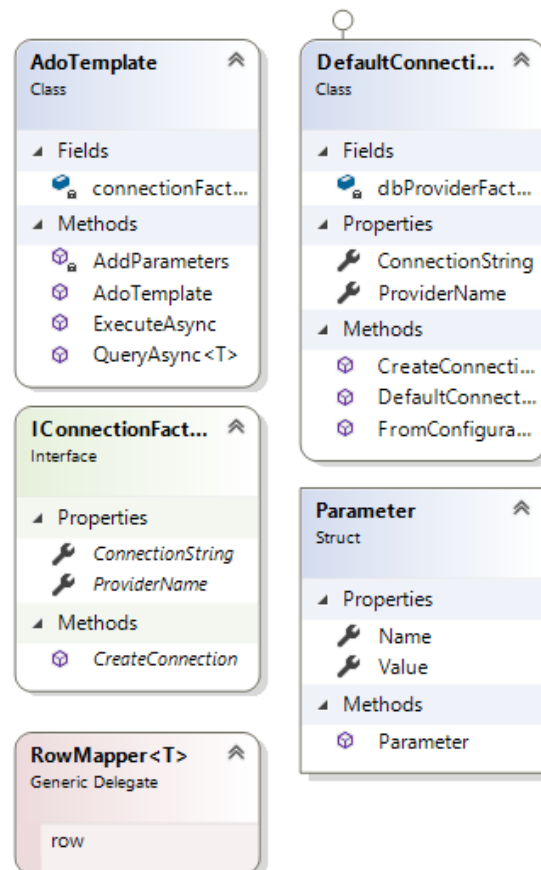


Abbildung 2: Common.Dal.Ado UML Diagramm

2.2 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>*.

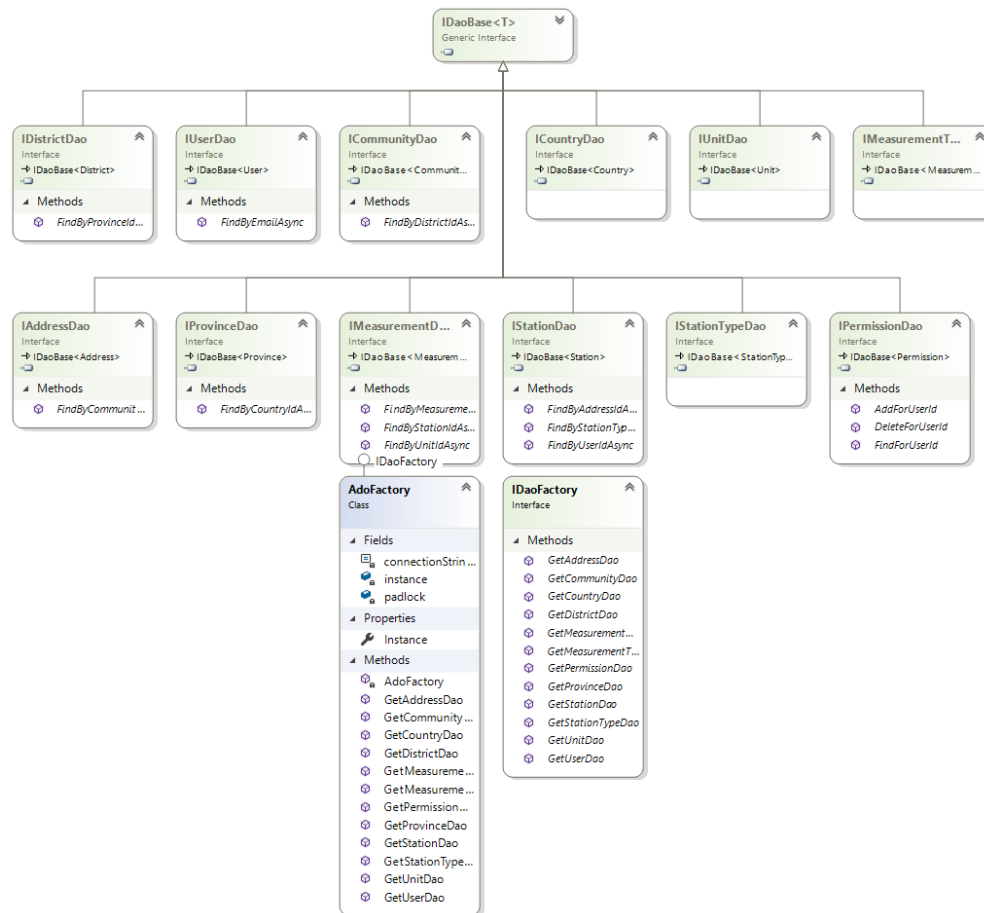


Abbildung 3: Wetr.Dal.Interface UML Diagramm

2.3 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.

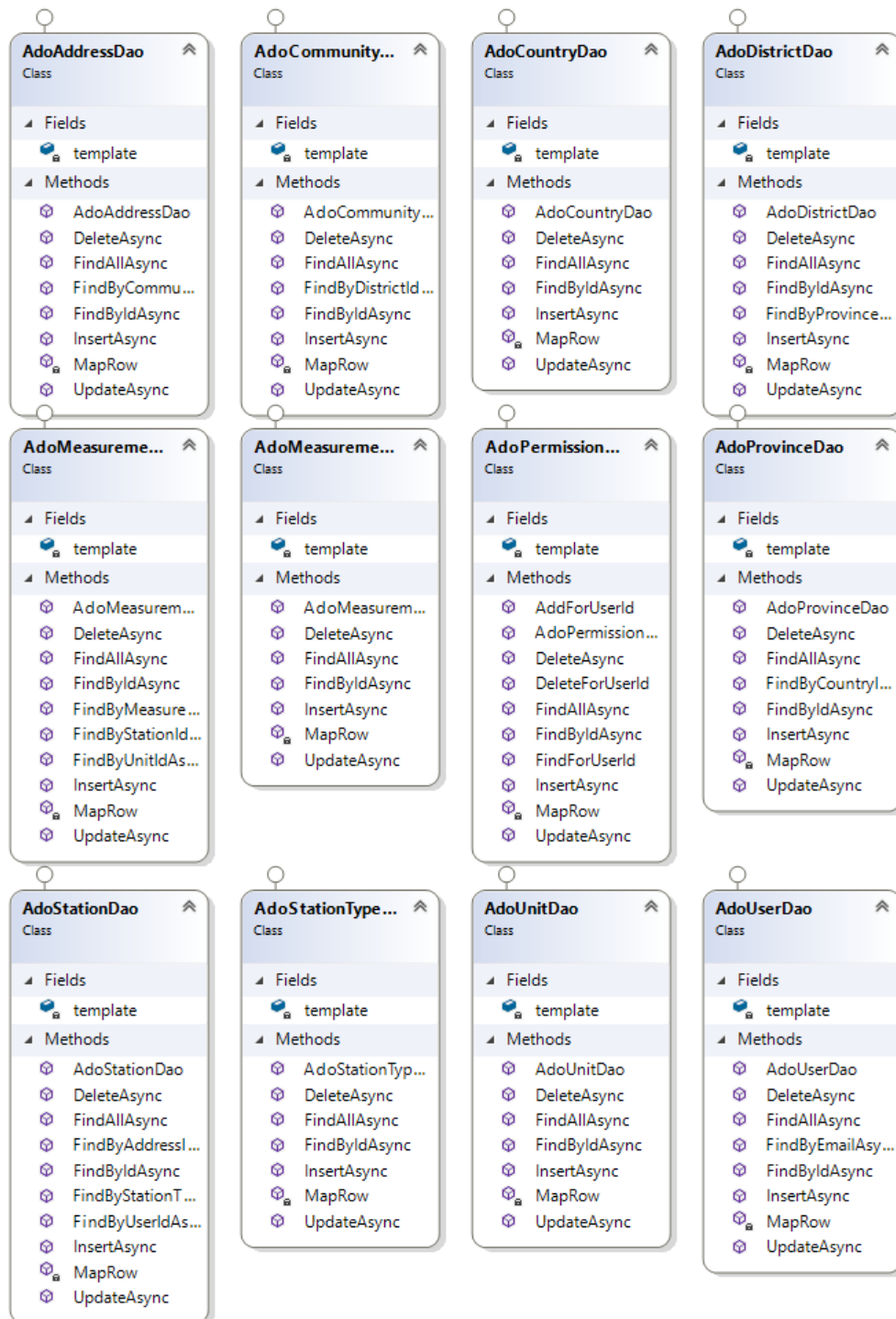


Abbildung 4: Wetr.Dal.Ado UML Diagramm

2.4 Wetr.Dal.Factory

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

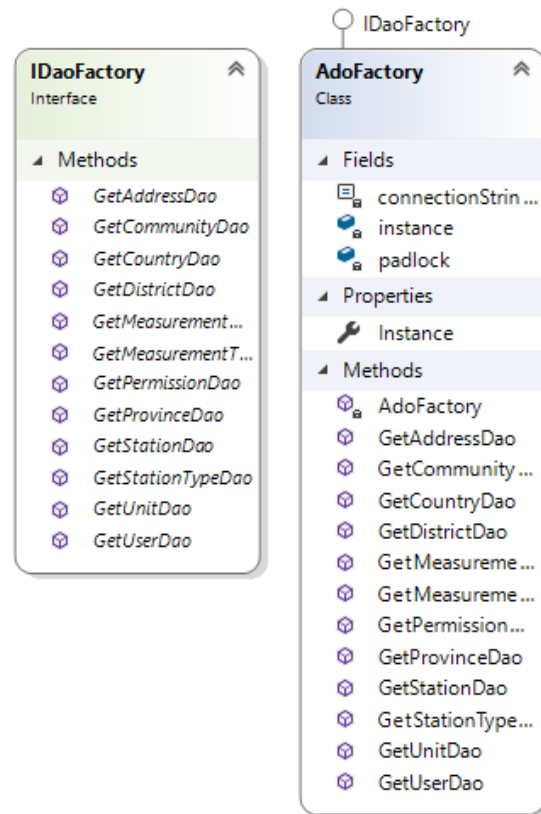


Abbildung 5: Wetr.Dal.Factory UML Diagramm

2.5 Wetr.Domain

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

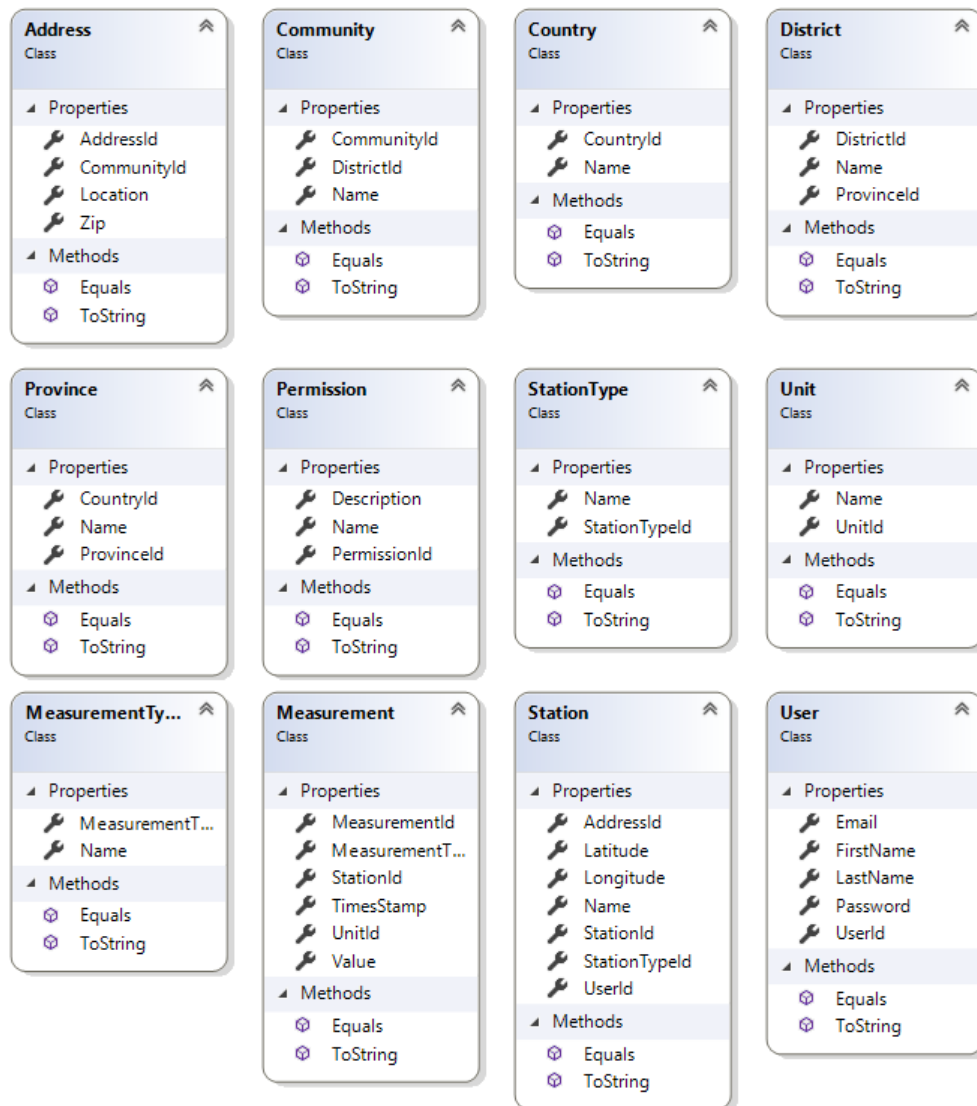


Abbildung 6: Wetr.Domain UML Diagramm

2.6 Wetr.Test.Dal

Beim *Wetr.Test.Dal* Projekt werden alle Funktionen von jedem *Dao* Objekt getestet.

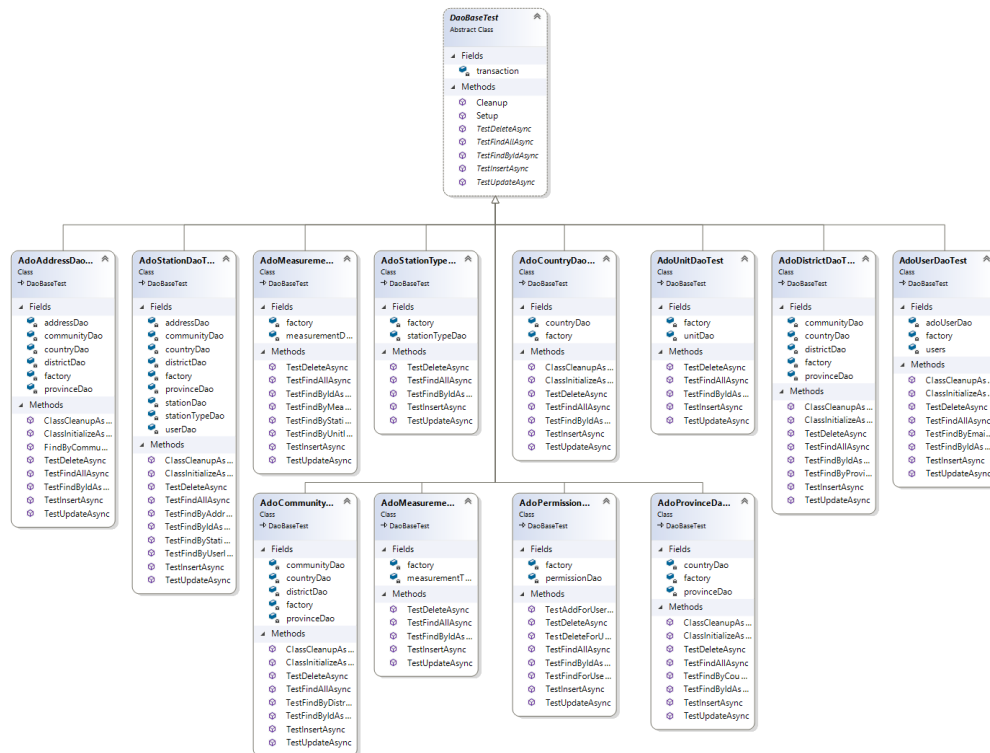


Abbildung 7: Wetr.Domain UML Diagramm

2.7 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-Insert⁵ 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 ± 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*

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

3 Installationsanleitung

3.1 Benötigte Programme und Voraussetzungen

Für dieses Projekt wird zusätzliche Software benötigt:

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

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
```

3.2 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.

⁶<https://www.docker.com/>

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

⁸<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';
```