Projektbezeichnung: Tytuł projektu:	[SmartRiver: Intelligentes Odergebiet/SmartRiver: Inteligentne Nadodrze]
Antragsnummer: Numer wniosku:	[85029892]
Output / Produkt:	Dokument: **„Datenerfassung und -verarbeitung"** / Dokument **„Gromadzenie i przetwarzanie danych"**

Betrifft: Aktivität 6 - **Datenerfassung und -verarbeitung**

Beschreibung der Aktivität: In Rahmen der Aktivität wird die Lösung für das Sammeln, Bearbeitung und Präsentation der Daten in dem System definiert, implementiert und getestet.

Die Autoren des Dokuments: IHP, UZ

Empfänger des Dokuments: Endanwender

Inhaltsübersicht

1. Einleitung
2. SmartDSM-Plattform
   2.1. Zugriffsrechte
   2.2. Operationen mit Daten
3. Szenarien
   3.1. Speichern von Netzwerkinformationen
   3.2. Übermittlung der Messergebnisse
   3.3. Übertragung von Steuersignalen
   3.4. Verarbeitung der Messergebnisse
4. Werkzeuge
5. Zusammenfassung
6. Fassungen des Dokuments

1. Einleitung

Eine der Anforderungen an das geplante System ist die Speicherung und Verarbeitung der von den Sensoren im überwachten Bereich erfassten Daten. Um diese Anforderung zu erfüllen, wurde die SmartDSM-Plattform verwendet, die aus einem Speicherteil (Middleware) und einem Verarbeitungsteil (Dienste) besteht. Die Implementierung der Datenspeicherung und -verarbeitung in der SmartDSM-Plattform ermöglicht die vollständige Kontrolle über die Struktur, die Verarbeitung und den Zugriff auf die Daten. Die Plattform bietet zusätzliche Tools für die Verwaltung, Sicherheit und Dienste zum Auffinden von Geräten und Daten im Netz. Die Plattform ist ein Vorschlag für die Umsetzung von Lösungen im Bereich der Smart City.

2. SmartDSM-Plattform

Die Plattform besteht aus Modulen, die zur Erstellung einer Lösung für die Datenspeicherung und -verarbeitung verwendet werden können. Es wurde in Java-Technologie entwickelt, die es ermöglicht, es im Kontext mehrerer Betriebssysteme laufen zu lassen. Die Sicherheit der Kommunikation wird durch eine Public Key Infrastructure (PKI) gewährleistet, die mit Hilfe der Java Secured-Socket Extension (JSSE) implementiert wurde. Jeder Kommunikationsteilnehmer hat ein Zertifikat, das zur Verschlüsselung von Nachrichten und zur Identifizierung des Teilnehmers verwendet wird. Die grundlegenden Module der Plattform sind ein Middleware-Server (Datenspeicherung) und eine Bibliothek (die die Data Interface Schnittstelle bietet), die die Kommunikation mit dem Server ermöglicht.

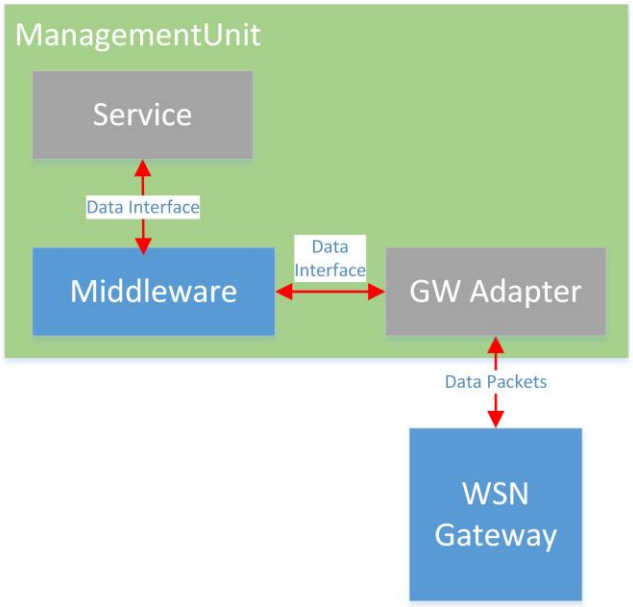

Abb. 1. Basismodule der SmartDSM-Plattform im Smart River-System

Die Systemfunktionalitäten, die auf der SmartDSM-Plattform basieren, werden als Dienste bezeichnet. Sie werden als separate Java-Anwendungen unter Verwendung der Data Interface-Bibliothek implementiert. Die Bibliothek stellt eine Verbindung zu einem ausgewählten Middleware-Server her und ermöglicht den Aufruf von Funktionen im Zusammenhang mit der Datenspeicherung (Schreiben, Lesen, Aktualisieren, Löschen) und der Kommunikation mit Plattformdiensten, wie z. B. anderen Middleware-Servern, Erkennung, Proxy und Teilnehmerverifizierung ("Certification Authority"). Die Bibliothek ermöglicht die Erstellung von so genannten Subscriptions, die es ermöglichen, Benachrichtigungen zu erhalten, wenn sich zum Beispiel ein ausgewählter Wert ändert oder ein erwarteter Wert gespeichert wird. Darüber hinaus ermöglicht die Bibliothek die Verwaltung von Berechtigungen (für die eigenen Daten oder für alle - mit der entsprechenden Berechtigung) auf einem ausgewählten Middleware-Server und die Abfrage des Status des Servers. Die Art und Weise, wie die Plattform implementiert ist, erlaubt den Einsatz mehrerer Hierarchieszenarien, je nach Anforderung kann ein zentraler Middleware-Server oder mehrere verteilte Server, die miteinander kommunizieren, eingesetzt werden.

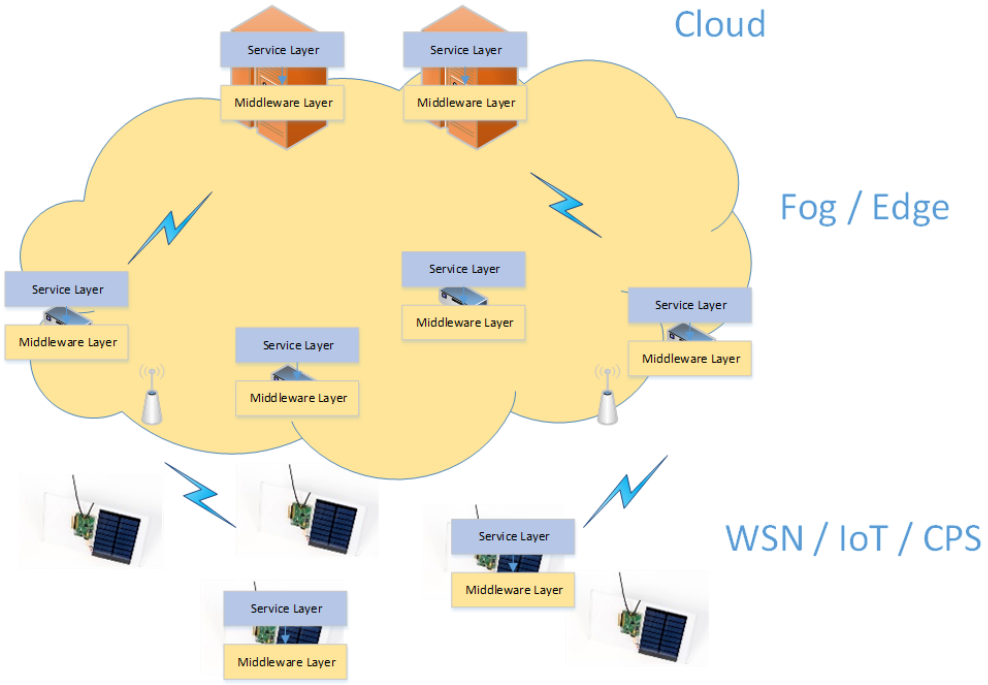

Abb. 2. Beispiel einer verteilten Netzstruktur

Datenoperationen werden auf der Grundlage von Variablen und deren Eigentümern durchgeführt. Eine Variable im Kontext der Plattform ist ein Objekt, das eine Liste von geordneten Wertefolgen (ein Tupel) definiert. Jede Variable hat einen Namen, Grenzwerte und Felddefinitionen. Für jede Variable kann der Eigentümer eine beliebige Anzahl von Werten (Tupel) speichern, die beliebige Daten darstellen können (z. B. Luftfeuchtigkeit, Temperatur usw.). Je nach Bedarf können die Tupel z.B. Steuerwörter zur Steuerung von Geräten im System, historische Werte von Messergebnissen oder Informationen über die Netzstruktur enthalten. Der Eigentümer im Kontext der Plattform ist die Person, die mit Hilfe von Diensten (die über die Data Interface Bibliothek mit dem Middleware-Server kommunizieren) Daten in Variablen speichert. Jeder Dienst handelt im Namen seines Eigentümers, in dessen Auftrag Aktionen auf dem Middleware-Server durchgeführt werden.

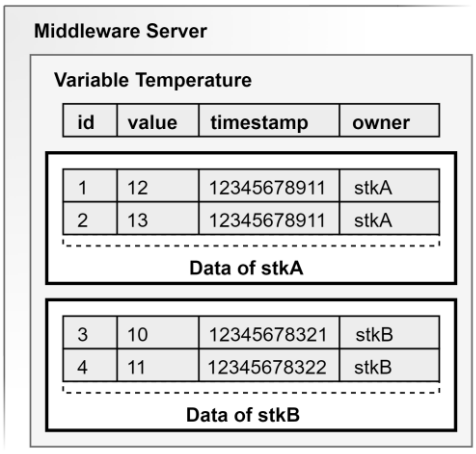

Abb. 3. Variable für die Speicherung von Temperaturmessungen

Abbildung 3 zeigt ein Beispiel für eine Variable, die die Temperaturergebnisse von zwei Besitzern (stkA und stkB) speichert. Auf dem Server gibt es eine Variable "Temperatur", in der alle Werte beider Besitzer gespeichert sind.

2.1. Zugriffsrechte

Standardmäßig hat nur der Eigentümer, in dessen Namen der Dienst das Ergebnis gespeichert hat, Zugriff auf die in der Variablen gespeicherten Werte. Der Dienst kann im Namen des Eigentümers über die Data Interface Bibliothek Zugang gewähren oder entziehen. Sobald der Zugriff gewährt wurde, können andere Benutzer (mit Hilfe von Diensten) z. B. Daten lesen oder Subscriptions erstellen, um Benachrichtigungen zu erhalten, wenn jemand mit der entsprechenden Berechtigung eine Operation an der Variablen durchführt (schreiben, lesen, aktualisieren, löschen). Tabelle 1 zeigt ein Beispiel für den Fall, dass ein beliebiger Dienst des Benutzers stkA den Temperaturwert des Benutzers stkB ohne zeitliche Begrenzung zwischen den Ablesungen ablesen kann. Abbildung 4 zeigt ein Beispiel für die Verwendung der Data Interface Bibliothek zur Erstellung der in Tabelle 1 aufgeführten Berechtigung.

Tabelle 1. Beispiel für die Berechtigung zum Lesen von Werten (ausgewählte Felder)

id	permission	owner	service	stakeholder	min_delay
1	variables.variable.Temperature .read	stkA	*	stkB	0

```
DataInterface dataIntf = <połączenie z serwerem middleware, jako stkA>
DataInterfaceRequest request = dataInf.createRequest();
String permission = "variables.variable.Temperature.read";
request.addPermission(permission, "stkA", "*", "stkB", ..., 0);
DataInterfaceResponse response = request.execute();
PolicyResponse policyResponse = response.getFirstWithType(PolicyResponse.class);

// Ergebnis in der Variable policyResponse
```

Abb. 4. Beispiel für eine Zugangsberechtigung (Data Interface Bibliothek)

Jeder Middleware-Server verfügt über einen Standardsatz von Berechtigungen, die jeden Besitzer berechtigen, grundlegende Operationen auf dem Server durchzuführen, z. B. Variablen zu erstellen, eine Liste von Berechtigungen oder Subscriptions herunterzuladen. Der Administrator kann die Berechtigungen frei ändern, einschließlich der Festlegung von Standardberechtigungen. Die Verwaltung erfolgt über die Webanwendung AdminGUI, die auf jedem Middleware-Server verfügbar ist.

2.2. Operationen mit Daten

Im Smart River-Projekt ist die Middleware-Schicht für die Speicherung und Verarbeitung von Daten zuständig. Bei den Daten kann es sich um Messergebnisse oder Steuersignale handeln. Je nach Art der Daten sind entsprechende Szenarien vorgesehen, nach denen die Datenübertragung, -verarbeitung und -speicherung erfolgt.

Die Änderung der Daten erfolgt durch den Aufruf der entsprechenden Methoden der Data Interface Bibliothek, die die Verbindung zum ausgewählten Middleware-Server herstellt.

```
DataInterface dataIntf = <połączenie z serwerem middleware, jako stkA>
DataInterfaceRequest request = dataInf.createRequest();
String permission = "variables.variable.Temperature.read";
request.addPermission(permission, "stkA", "*", "stkB", ..., 0);
String[] keys = new String[] { "accuracy" };
Object[] values = new Object[] { 0.5 };
request.write("Temperature", "stkA”, <serwer>, 22, keys, values).as("write");
request.read("Temperature", "stkB", <serwer>, <filtr>).as("read");
DataInterfaceResponse res = request.execute();
VariableResponse wr = res.getFirstNamedAs("write", VariableResponse.class);
Class<VariablePageResponse> rr_class = VariablePageResponse.class;
VariablePageResponse rr = res.getFirstNamedAs("read", rr_class);

// Ergebnis in Variablen wr und rr
```

Abb. 5. Beispiel für das Schreiben und Lesen von Daten

Die Methoden, die mit dem Lesen von Daten und dem Abrufen einer Liste von Objekten (Variablen, Berechtigungen, Abonnements) verbunden sind, können seitenweise angeordnet, gefiltert und sortiert werden, so dass alle Werte in der gewählten Reihenfolge abgerufen werden können.

Um Benachrichtigungen zu erhalten, wenn die Werte der Variablen von anderen Besitzern geändert werden, ist es erforderlich, die Schnittstelle SubscriptionListener zu implementieren, einen Verweis auf die Instanz Data Interface zu übergeben und eine Subscription auf dem Middleware-Server zu erstellen. Abbildung 6 zeigt, wie die Subscription auf der SmartDSM-Plattform funktioniert. Zunächst wird ein Data Interface Objekt erstellt, dann wird ein Objekt vom Typ SubscriptionListener übergeben. Schließlich wird ein Abonnement erstellt. Sobald das Abonnement erstellt wurde und der Eigentümer Zugriff auf die Daten hat, für die das Abonnement erstellt wurde, wird die Methode onSubscription aufgerufen, sobald ein Wert in eine Variable geschrieben wird. Wenn ein Filter übergeben wurde, wird die Methode nur aufgerufen, wenn der Wert die Anforderungen des übergebenen Filters erfüllt.

```
DataInterface dataIntf = DataInterface.create(...)
.registerSubscriptionListener(new SubscriptionListener() {
  public void onSubscription(SubscriptionInfo info) {
    System.out.println("Otrzymano powiadomienie: " + info);
  }
})
DataInterfaceRequest request = dataInf.createRequest();
String permission = "variables.variable.Temperature.read";
request.subscribe(VariableEvent.WRITE, <nazwa zmiennej>, <nazwa właściciela>,
<serwer docelowy>, <filtr>);
DataInterfaceResponse response = request.execute();
SubscriptionResponse sr = response.getFirstWithType(SubscriptionResponse.class);

// Ergebnis der Erstellung einer Subscription in Variable sr
```

Abb. 6. Erstellen von Subscriptions und Erhalten von Benachrichtigungen

3. Szenarien

Abbildung 7 zeigt die Systemarchitektur des Smart-River-Projekts, auf das sich die Abschnitte 3.1 bis 3.3 beziehen. Die Datenquellen sind WSN-Knoten, die über Sensoren für jeden überwachten Wert im System verfügen. Die Messergebnisse werden auf Middleware-Servern gespeichert. Der übergreifende Middleware-Server (SuperUnit) stellt die Webanwendung den Endnutzern über das Internet zur Verfügung. Dienste wie die Ermittlung (Discovery) und die indirekte Kommunikation (Proxy) sind im Internet verfügbar.

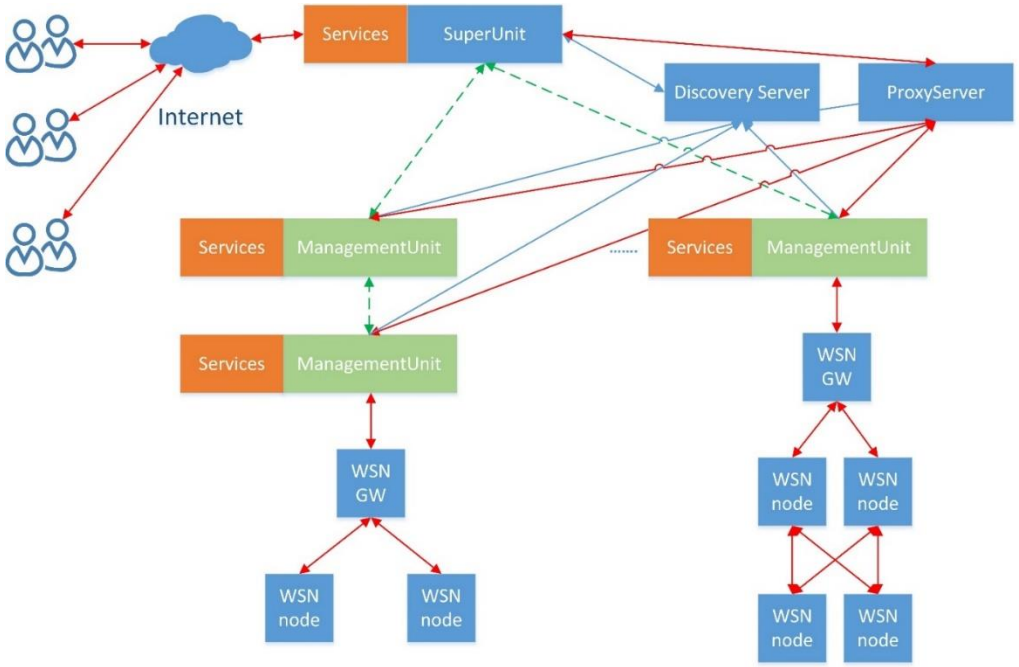

Abb. 7. Architektur des Messsystems

3.1. Speichern von Netzwerkinformationen

Abbildung 8 zeigt die Struktur des Smart River-Netzwerks, die in Variablen auf dem Middleware-Server gespeichert ist. Die Variable "Networks" enthält Informationen über die im System enthaltenen Netze (Frankfurt und Słubice). Jedes Netz hat eine Liste von Knoten (Nodes). Jeder Knoten verfügt über Standortinformationen (LocationNames), Knotentypinformationen (NodeTypeNames) und eine Liste von Geräten (NodeTypes). Jedes Gerät verfügt über Gerätetypinformationen (DeviceTypeNames) und eine Liste von Sensoren (SensorTypes).

Im Rahmen des Projekts wurde ein Tool entwickelt (Abschnitt 4), mit dem die Konfiguration heruntergeladen, aktualisiert, exportiert und visualisiert werden kann.

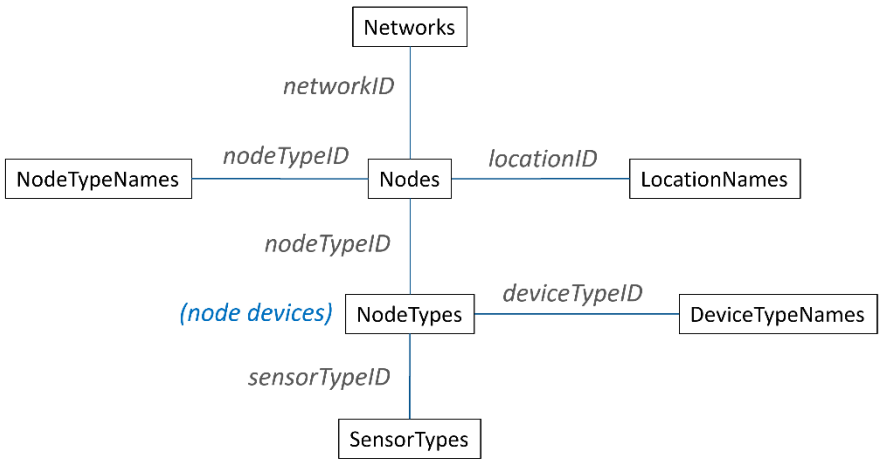

Abb. 8. Beziehungen zwischen den Variablen, die Informationen über die Struktur des Systems speichern

3.2. Übermittlung der Messergebnisse

Abhängig von den gemessenen Parametern senden die WSN-Knoten die Messergebnisse an das Node-Gateway (WSN GW) entsprechend der Frequenz für den aktuellen Systembetriebsmodus (Normal, Test, Alarm, Test und Service). Anschließend senden die GW-Knoten des WSN die Messergebnisse an die für den Betrieb des GW-Knotens zuständigen Dienste. Die Services speichern die Ergebnisse auf Middleware-Servern (Abschnitt 2.2). Jeder Messwert wird in einer separaten Variablen gespeichert, die zusätzliche Felder zur Beschreibung des Messergebnisses enthält - den geografischen Standort, die Knoten-ID, die Geräte-ID und den Sensor, der die Messung vorgenommen hat.

Tabelle 2. Beispiel Ergebnis einer Messung

id	timestamp	owner	source	value	nodeId	deviceTypeId	sensorId	lat	lon
1	1653909134713	Miasto1	<Quelle>	20	1	1	1	54.312	14.333

3.3. Übertragung von Steuersignalen

Die Konfigurations- und Steuersignale werden in Variablen auf dem Middleware-Server gespeichert. Die Übertragung erfolgt, indem der nächste Wert in die Variable geschrieben wird. Server, die sich auf die Konfiguration stützen oder Operationen auf der Grundlage von Signalen durchführen, erstellen Subscriptions, so dass der neue Wert mitgeteilt werden kann, wenn er verfügbar ist.

Tabelle 3. Beispiel Variable mit einem Steuerwert

id	timestamp	owner	source	value	name	nodeId	deviceTypeId	sensorId
1	1653909134713	Miasto1	<Quelle>	30000	interval	1	1	1
2	1653909149451	Miasto1	<Quelle>	true	ABC_enabled	1	1	1

3.4. Verarbeitung der Messergebnisse

Verarbeitungsdienste erstellen Subscriptions und erhalten sofort Benachrichtigungen, wenn neue Messergebnisse erfasst werden. Die Ergebnisse der Verarbeitung werden in Variablen gespeichert, für die weitere Subscriptions erstellt werden können, um Benachrichtigungen über verarbeitete Messergebnisse zu erhalten.

4. Werkzeuge

Um die Netzwerkkonfiguration einfacher zu verwalten, wurde ein Tool entwickelt, das die aktuelle Konfiguration vom Middleware-Server abruft, Änderungen vornimmt, exportiert und visualisiert. Das Tool verwendet die Data Interface Bibliothek zur Kommunikation mit dem Server, auf dem die Konfiguration gespeichert ist.

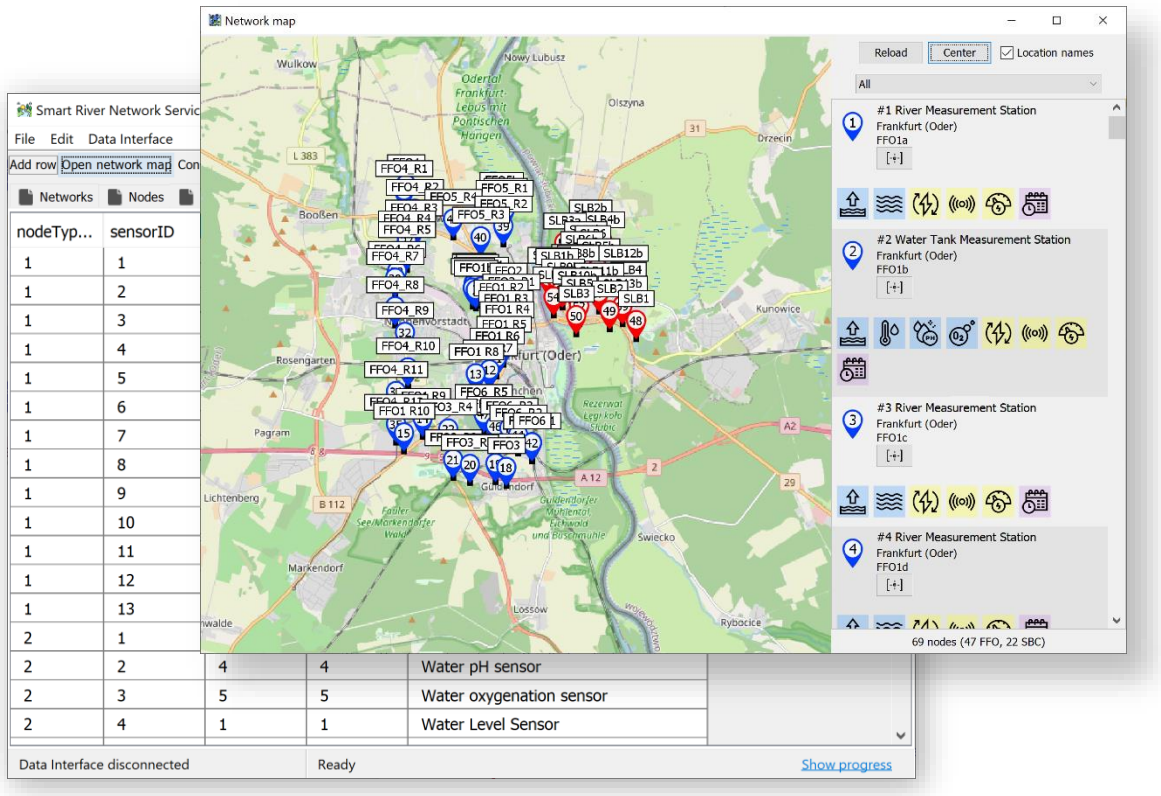

Abb. 9. Die Software zur Verwaltung der Netzwerkkonfiguration

5. Zusammenfassung

In dem Dokument wird dargelegt, wie die Daten in dem geplanten System erfasst und verarbeitet werden. Die SmartDSM-Plattform wird vorgestellt, und es wird erläutert, wie die Plattform im Projekt verwendet wird. Die Netzstruktur und die Methodik, die bei der Bearbeitung der verschiedenen Datentypen (Messergebnisse, Konfiguration, Steuersignale) angewandt wird, werden vorgestellt. Die Verarbeitung von Messergebnissen und deren gemeinsame Nutzung wird vorgestellt. Darüber hinaus wird ein Tool zur Konfiguration und Visualisierung des Netzes vorgestellt.

Die Plattform und die Art und Weise, wie sie genutzt wird, ermöglichen es, das System in Zukunft problemlos um neue Funktionen zu erweitern.

6. Fassungen des Dokuments

Geändert von	Änderung	Ausgabe	Version
IHP	Initiale Version mit Änderungen	A	- (0)
		A	1