



OPC-UA

ANDROID CLIENT

Christian Cavallo & Noemi Buggea

1 SOMMARIO

2	Introduzione	2
2.1	Requisiti	2
2.2	Descrizione progetto	3
3	Implementazione progetto.....	3
3.1	Discovery	4
3.2	Browse	5
3.3	Read	7
3.4	Write	8
3.5	Subscribe	9
3.6	Creazione di oggetti personalizzati.....	10
3.7	Visualizzazioni delle subscriptions.....	11

2 INTRODUZIONE

OPC UA rappresenta il principale erede dello standard OPC Classic e uno dei suoi vantaggi principali è l'indipendenza dalla piattaforma, che ne consente l'integrazione facile in Windows, Linux, Mac, Android e altre piattaforme, un aspetto molto importante per il settore manifatturiero in cui le macchine e i sistemi spesso funzionano su piattaforme differenti. Inoltre, tale standard combina tutti i distinti protocolli in un'unica specifica ai fini della semplificazione. Tra gli altri vantaggi chiave si annoverano il fatto che lo standard OPC UA può essere implementato con facilità, lo scambio di dati è sicuro, è compatibile con i sistemi legacy e le infrastrutture esistenti, e consente la scalabilità.

Un OPC UA Client consente di effettuare diverse operazioni (lettura, scrittura, publishing o subscribing) su un OPC UA Server. In particolare è possibile creare sottoscrizioni alla quale aggiungere dei Monitored Items, utili ai fini di osservare/monitorare un particolare nodo.

Gli impianti controllati mediante sistemi SCADA presentano un modello grafico affine all'ambito industriale. A tal proposito, lo scopo del progetto consiste nel realizzare un app Android che implementi lo standard OPC UA e proponga un'interfaccia grafica in stile SCADA.

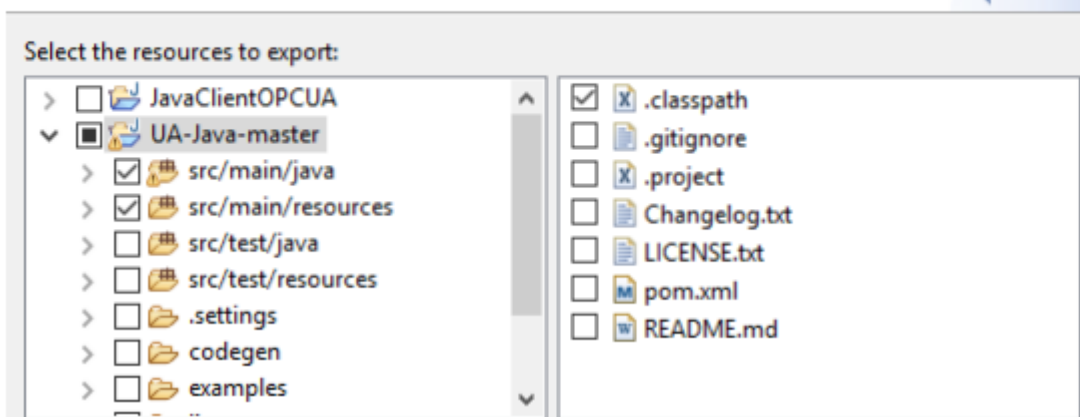
2.1 REQUISITI

Per poter implementare l'OPC UA Java stack nell'IDE Android è necessario compilare il source dall'apposita repository (<https://github.com/OPCFoundation/UA-Java-Legacy>) ed esportare un package jar.

A seconda dell'IDE utilizzato, la procedura consiste nel clonare la repository sopra descritta ed effettuare l'export delle principali cartelle contenute in "src":

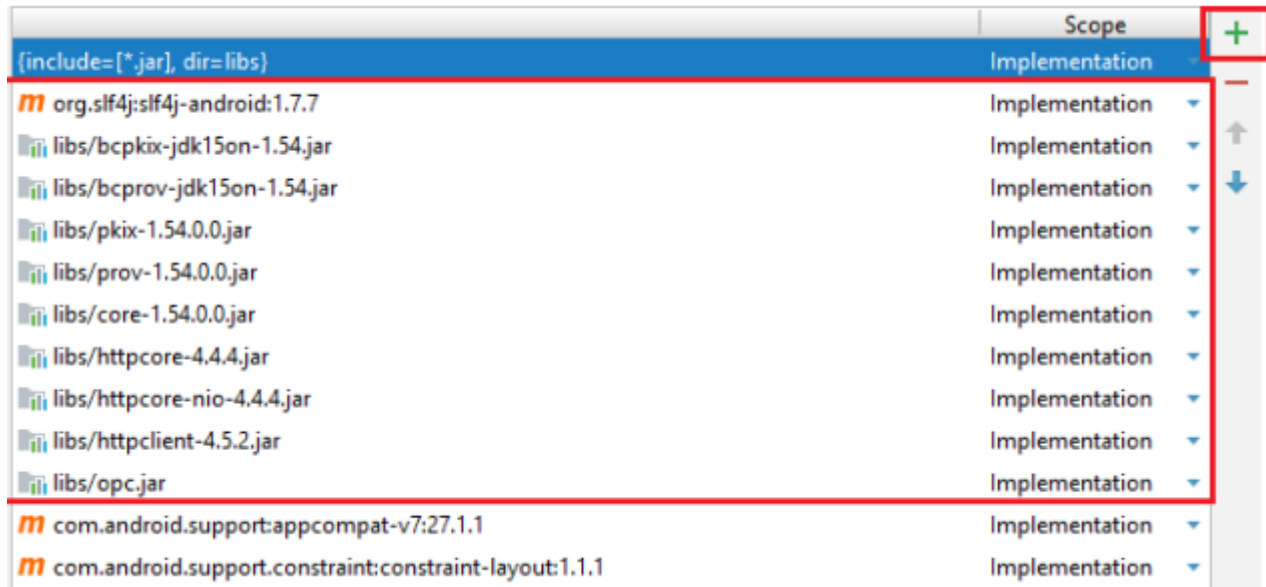
JAR File Specification

Define which resources should be exported into the JAR.



I file jar risultanti devono poi essere importati come librerie nel progetto corrente android.

La libreria necessita inoltre di alcune dipendenze esterne. Vengono qui riportate in ordine di compilazione:



	Scope
(include=[*.jar], dir=libs)	Implementation
org.slf4j:slf4j-android:1.7.7	Implementation
libs/bcpkix-jdk15on-1.54.jar	Implementation
libs/bcprov-jdk15on-1.54.jar	Implementation
libs/pkix-1.54.0.0.jar	Implementation
libs/prov-1.54.0.0.jar	Implementation
libs/core-1.54.0.0.jar	Implementation
libs/httpcore-4.4.4.jar	Implementation
libs/httpcore-nio-4.4.4.jar	Implementation
libs/httpclient-4.5.2.jar	Implementation
libs/opc.jar	Implementation
com.android.support:appcompat-v7:27.1.1	Implementation
com.android.support.constraint:constraint-layout:1.1.1	Implementation

2.2 DESCRIZIONE PROGETTO

Il progetto consiste nella realizzazione di un'applicazione Android che implementa sia lo stack OPC UA Java, sia una UI in stile SCADA con elementi animati.

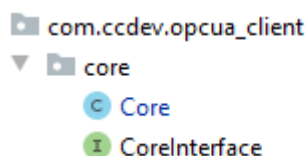
Tra i principali elementi utilizzati nelle rappresentazioni di impianti, abbiamo preso in considerazione i seguenti:

- Pompa
- Valvola
- Recipiente
- Sensore (Temperatura, pressione, umidità e generico)

Ognuno di essi dispone di due differenti modalità di visualizzazione (progress bar o indicatore analogico). Infine abbiamo aggiunto la possibilità di visualizzare le variazioni dei dati su un grafico aggiornato costantemente.

3 IMPLEMENTAZIONE PROGETTO

Genericamente, l'app è composta da un Core principale istanziato come Singleton che contiene i wrappers ai principali comandi esposti dallo stack (Read, Write, Discovery, Connect, ecc..).



```
com.ccdev.opcua_client
└── core
    ├── Core
    └── CoreInterface
```

È stato fondamentale utilizzare un singleton in modo tale da dare accesso alla OPC UA session da qualunque fragment. È inoltre presente un'interfaccia "CoreInterface" utilizzata per implementare l'observation pattern che consente alle classi registrate di essere notificate ogni qualvolta si presenta un aggiornamento sui dati.

3.1 DISCOVERY

La prima schermata dell'app (Figura 1), consente all'utente di effettuare il Discovery dell'OPC UA Server sulla base di un URL. Gli endpoints proposti sono filtrati per "opc.tcp" ed in ordine di sicurezza. Cliccando su un endpoint verrà "creata" la sessione ma non attivata. Un successivo Dialog (Figura 2), consente di scegliere la modalità di autenticazione tra anonima e username/password, eseguendo l'attivazione effettiva della sessione.

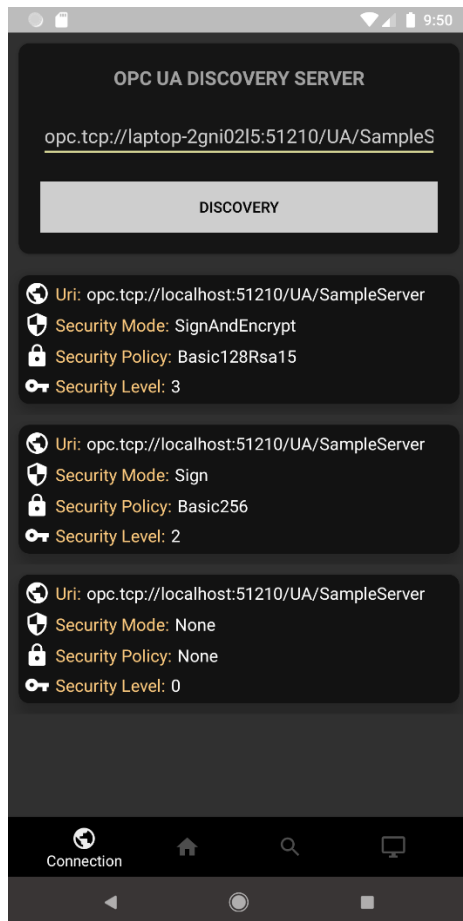


Figura 1

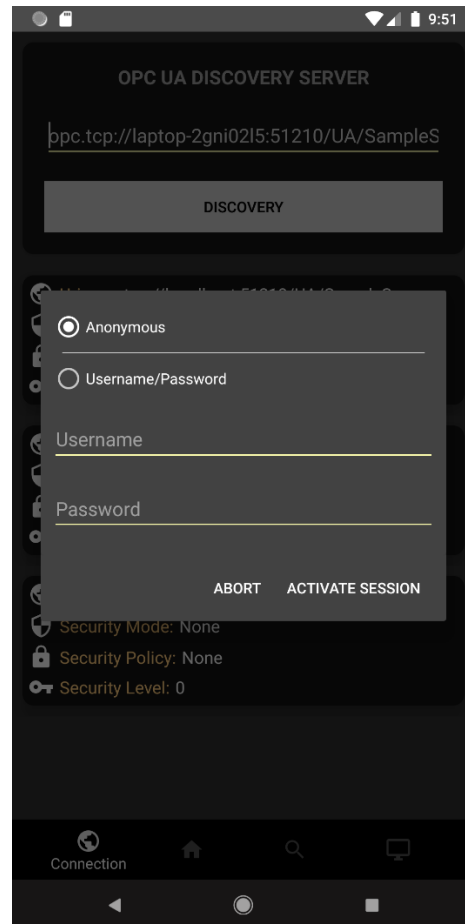


Figura 2

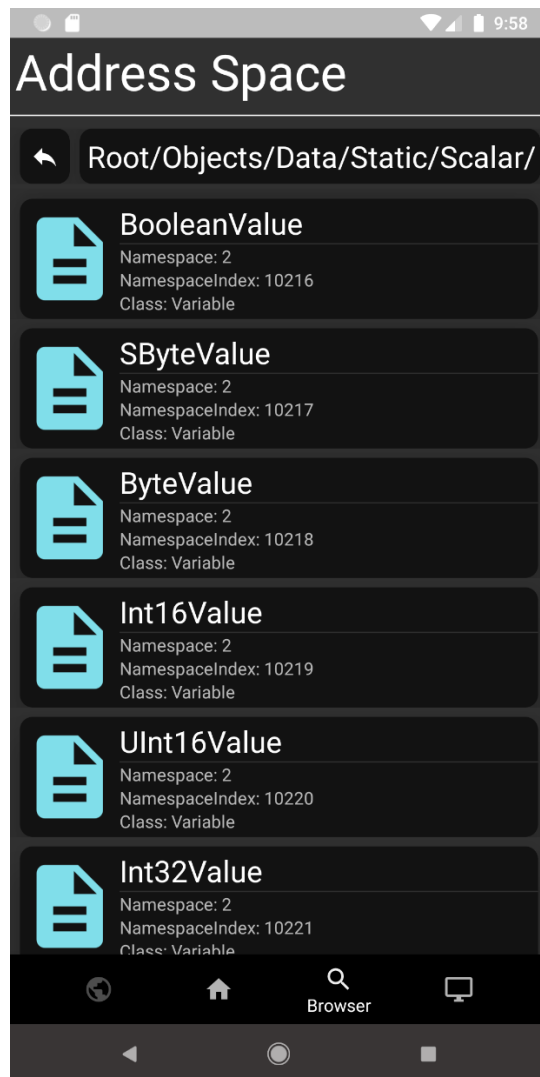
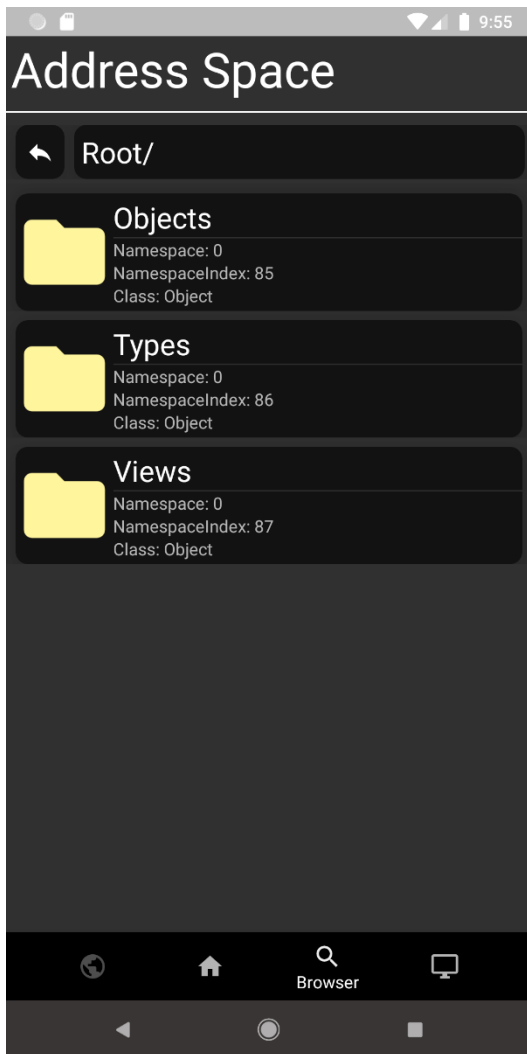
Il sorgente che si occupa di gestire questo fragment è il seguente:

```
ui
├── adapters
├── browser
└── connection
    └── ConnectionFragment
```

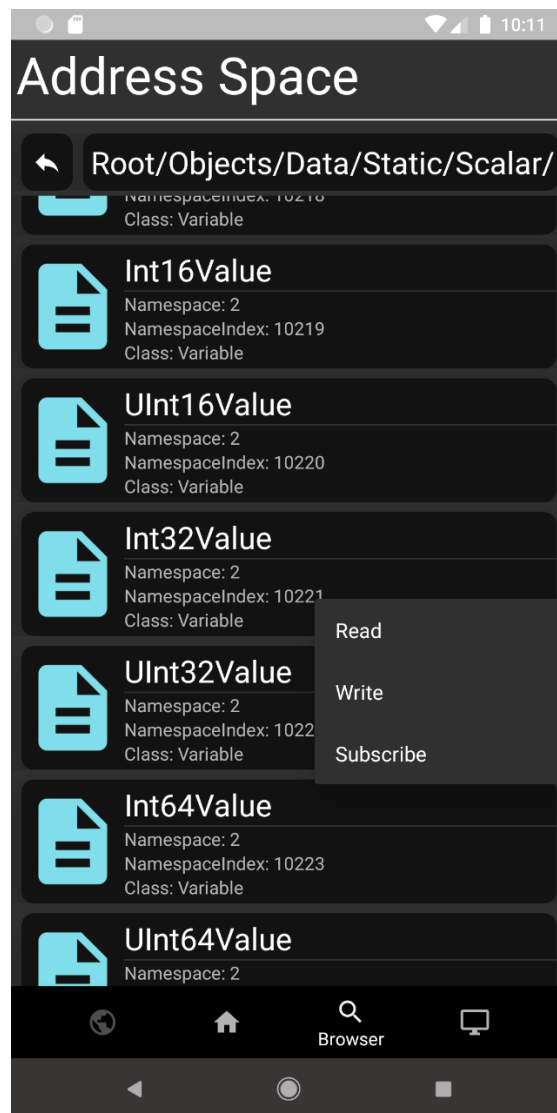
3.2 BROWSE

Il fragment “Browse” consente di navigare all’interno dell’address space dell’OPC UA Server. È stata implementata un’interfaccia user friendly con navigation bar e comandi di navigazione.

La browse filtra esclusivamente i nodi il cui NodeClass è “Variable” o “Object”. Non vengono visualizzati altre tipologie di nodo tra cui metodi e reference.



Una volta raggiunto il nodo desiderato, basta effettuare un tocco prolungato ed un menu contestuale mostrerà le possibili azioni tra: Read, Write, Subscribe.



Il sorgente legato al Browse Fragment è il seguente:

```
ui
├── adapters
├── browser
│   └── BrowserFragment
```

3.3 READ

La funzionalità di read è stata implementata tramite una dialog dove è possibile specificare i parametri relativi da una richiesta di Read, ovvero MaxAge e TimeStamps. Infine basta solo cliccare su “Read” per richiedere il dato relativo al nodo.

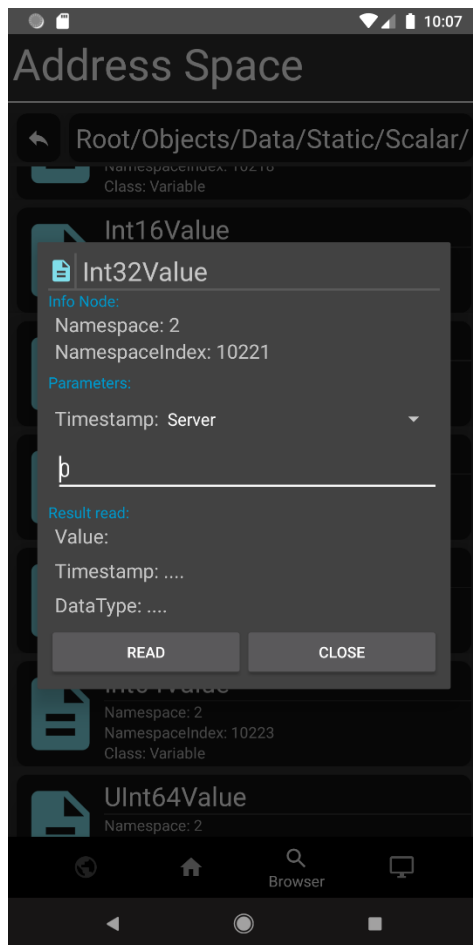


Figura 3: Lettura ancora da effettuare

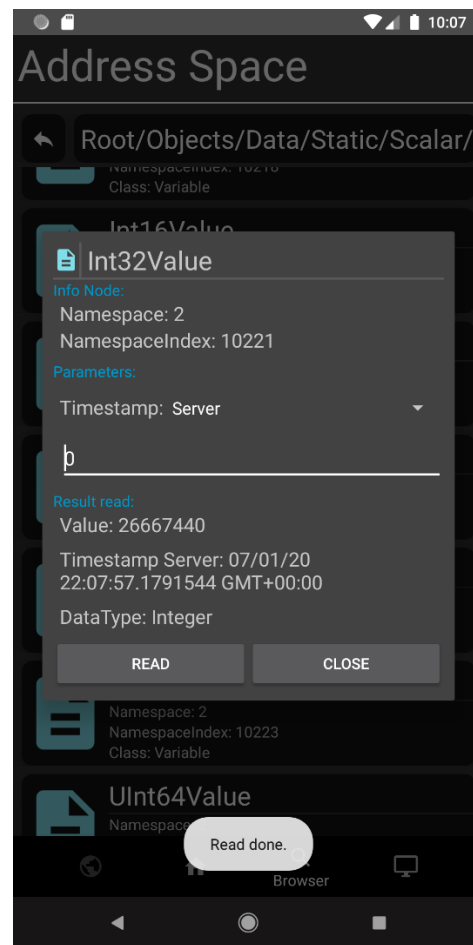


Figura 4: Lettura effettuata

I metodi sono contenuti all'interno del Browse Fragment e delimitati da degli spacer commentati:

```
903 //Read =====
904
905 TextView nodeValueView;
906 TextView nodeTimestampView;
907
908 String currentNodeDataType;
909 TextView dataTypeView;
910
911 private void ShowReadDialog() {...}
912
913 private void ReadNode(final Double maxAge, final TimestampsToReturn timestamps) {...}
914
915 //=====
```


3.4 WRITE

L'implementazione grafica della Write è identica alla Read, ovvero tramite un Dialog. Per una corretta scrittura del nodo è necessario conoscere il tipo di dato che esso contiene. A tal fine viene prima effettuata una read, la quale contiene l'informazione cercata.

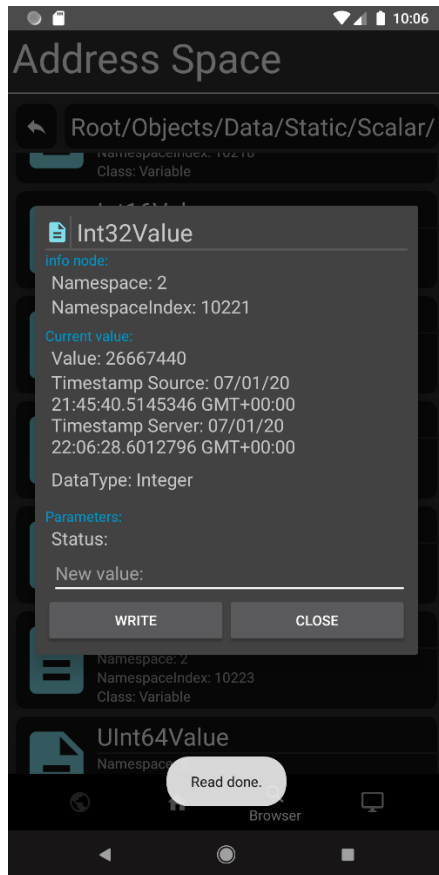


Figura 5:

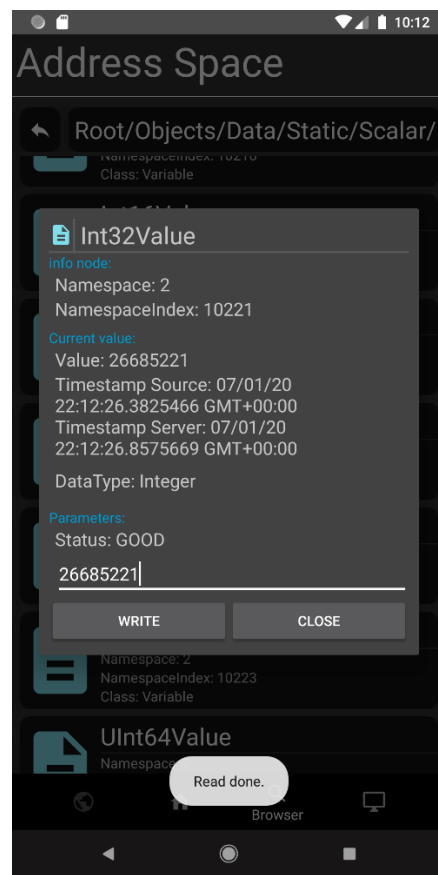


Figura 6: Scrittura effettuata

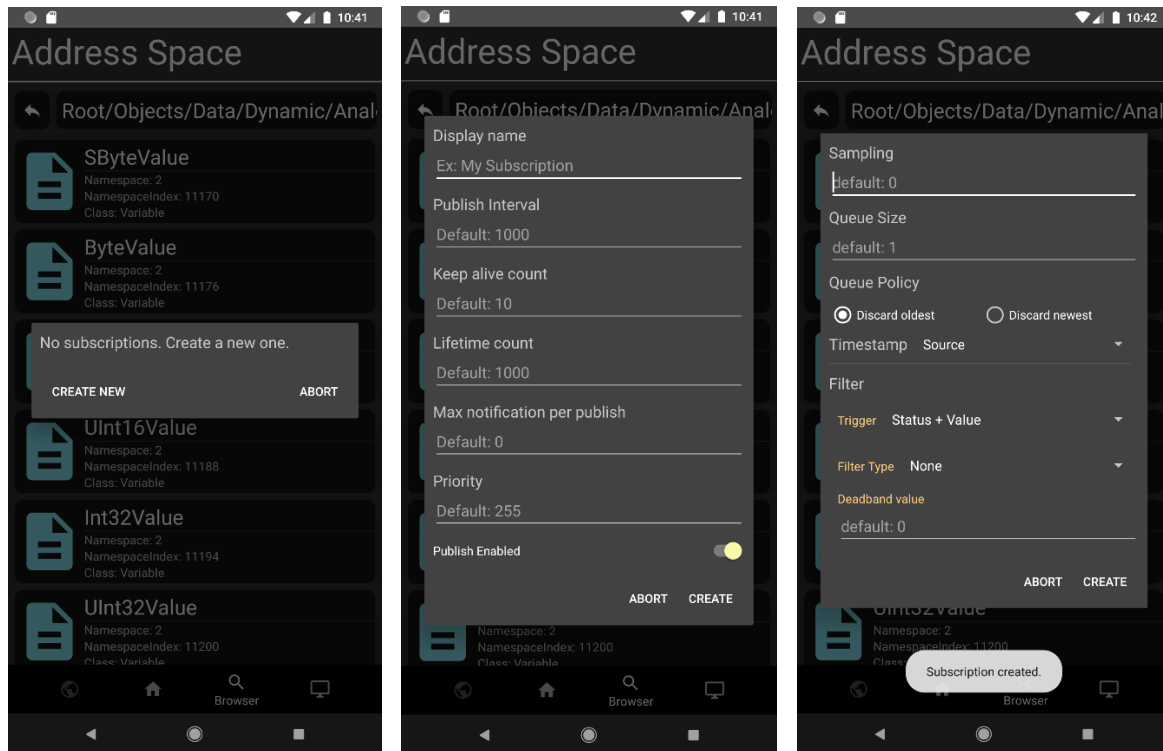
Inoltre ad ogni scrittura, viene ripetuta una lettura come forma di verifica (non necessaria se consideriamo lo Status Code ritornato dalla richiesta di write) dell'effettiva scrittura del dato.

Il codice di questa funzionalità è contenuto all'interno di BrowserFragment e delimitato da degli spacers commentati:

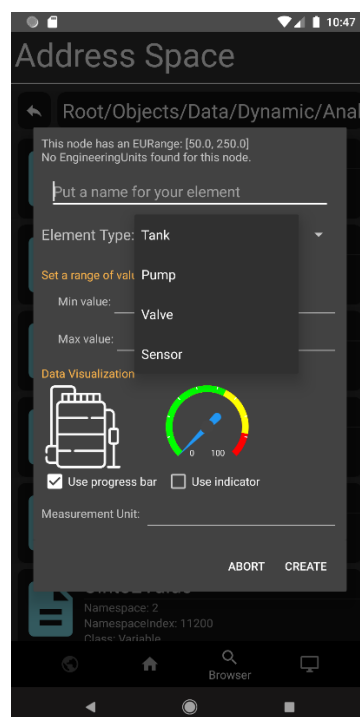
```
1053 //Write =====
1054
1055 private void ShowWriteDialog() { ... }
1178
1179 private void WriteNode(final WriteRequest req) { ... }
1219
1220 // =====
```

3.5 SUBSCRIBE

Questa funzionalità ci consente di creare/selezionare una subscription con il fine di inserirvi un Monitored Item legato al nodo selezionato. Consiste in una sequenza di Dialogs che ci consentono di impostare i parametri fondamentali per una subscription inizialmente, per poi passare alla creazione del Monitored Item.



Al termine della creazione del Monitored Item verrà richiesto se si vuole collegare un oggetto personalizzato a quest'ultimo. Ciò permette di modellare il Monitored Item in una componente tipica dei sistemi SCADA (tra cui Valvola, Pompa, Recipiente, Sensore).



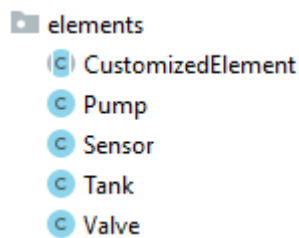
3.6 CREAZIONE DI OGGETTI PERSONALIZZATI

Per poter associare un'interfaccia di visualizzazione personalizzata ad un Monitored Item, è necessario prima definire un modello di oggetto. Il modello utilizzato è così composto:

```
public abstract class CustomizedElement {  
  
    public enum VisualizationType {  
        PROGRESS_BAR,  
        INDICATOR,  
        BOTH  
    }  
  
    ExtendedMonitoredItem monitoredItem;  
    String name;  
    String unit;  
    VisualizationType visualization;  
}
```

Contiene un nome, un'unità di misura, il MonitoredItem assegnato ed una tipologia di visualizzazione dei dati.

Trattandosi di una classe astratta, lo scopo consiste nella creazione di una gerarchia di classi figlie rappresentanti le singole componenti:



Ciascuna di essa presenta inoltre 2 campi aggiuntivi per il range dei valori ammissibili.

Dunque, tornando alla creazione del Customized Element, è necessario ricercare l'eventuale presenza di due nodi fondamentali "EuRange" ed "EngineeringUnits" che contengono rispettivamente il range di valori ammissibili e l'unità di misura utilizzata. Nel caso questi nodi non siano presenti, il dialog permette l'inserimento manuale di questi valori, risolvendo quindi il problema.

Le due funzioni di ricerca del range e dell'unità di misura sono definite all'interno del BrowserFragment:

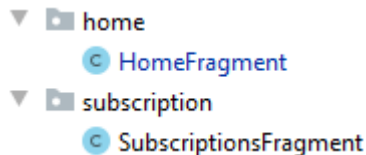
```
254  @  private Range searchEuRange(ReferenceDescription r) {...}  
272  
273  @  private EUInformation searchMeasurementUnit(ReferenceDescription r) {...}
```

Dopodiché è necessario selezionare almeno una modalità di visualizzazione dei dati tra "Progress bar" ed "Indicator".

Confermando il tutto, l'oggetto viene creato e legato al Monitored Item corrente.

3.7 VISUALIZZAZIONI DELLE SUBSCRIPTIONS

Sono state implementate due Fragment con differenti visualizzazioni delle subscriptions create. Ricordando che ogni subscription “contiene” dei monitored items e che quest’ultimi potrebbero a sua volta essere collegati a dei CustomizedElements, è stato fondamentale implementare una prima fragment che visualizzasse chiaramente la struttura tra Subscriptions e Monitored Items (SubscriptionFragment), ed una seconda fragment che invece mostrasse i CustomizedElements associati ai Monitored Items (HomeFragment).



Iniziamo considerando il “SubscriptionsFragment”. Esso propone una visualizzazione ad elenco delle subscriptions che a sua volta propongono come sotto elenco la lista dei Monitored Items creati. Grazie all’observation pattern inizialmente citato, il fragment è aggiornato sui valori correnti di ciascun Monitored Item. È possibile inoltre effettuare delle operazioni di Start/Stop sull’intera subscription o sul singolo oggetto. Infine è stata implementato un dialog con la funzionalità di monitoring sugli ultimi valori ricevuti sul Monitored Item selezionato:

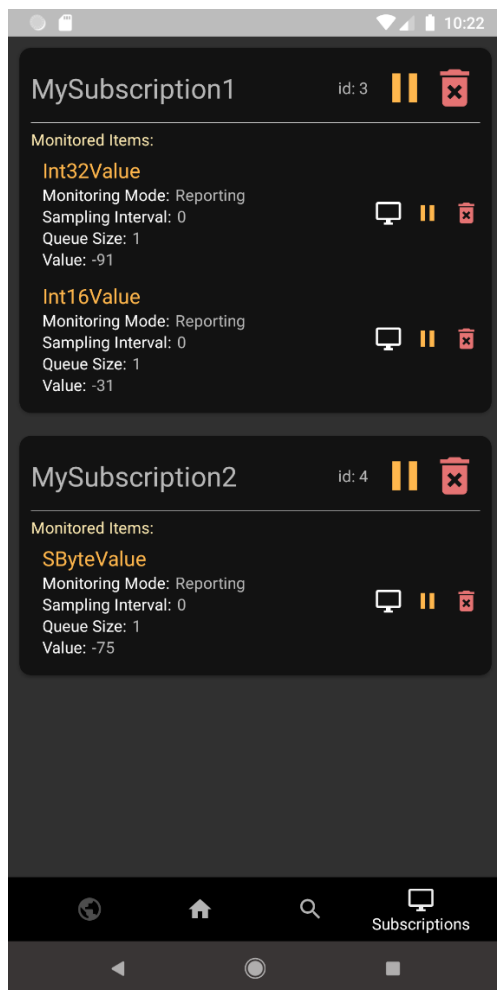


Figura 7:

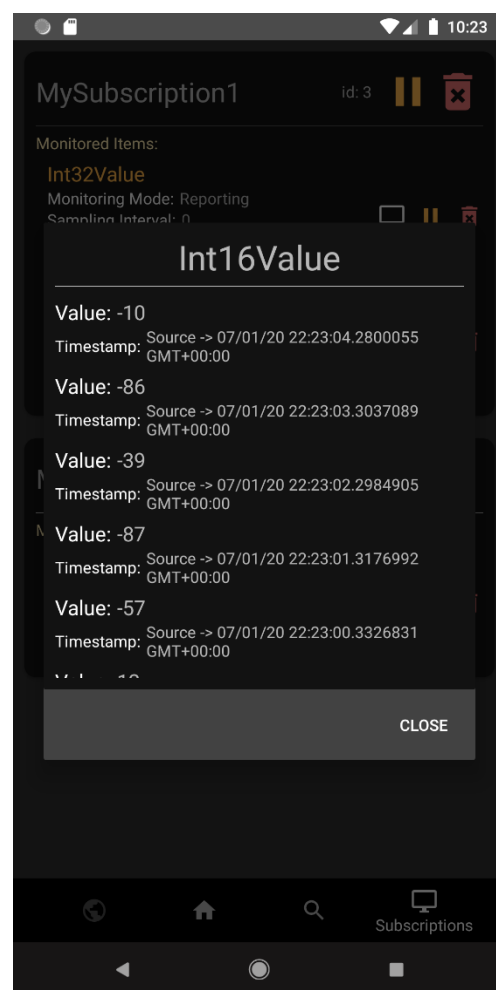


Figura 8:

Il fragment “HomeFragment” invece propone la visualizzazione “SCADA” applicata ai Monitored Items mediante i CustomizedElements. Sono stati definiti dei layouts che modellano quest’ultimi per poi essere inseriti in una vista ad elenco.

A seconda della modalità di visualizzazione dei dati selezionata al momento della creazione del Customized Element possiamo osservare la presenza di Progress Bar o Indicators. Anche in questo caso i dati sono aggiornati all’arrivo di nuovi valori da parte del server.

Infine è stata implementata una differente modalità di Monitoring per questi elementi, ovvero un grafico con assi cartesiani in cui viene effettuato il drawing dei valori ricevuti per il Monitored Item associato. Sull’asse delle ascisse abbiamo il tempo, mentre sulle ordinate abbiamo il tipo di dato relativo al monitored item.

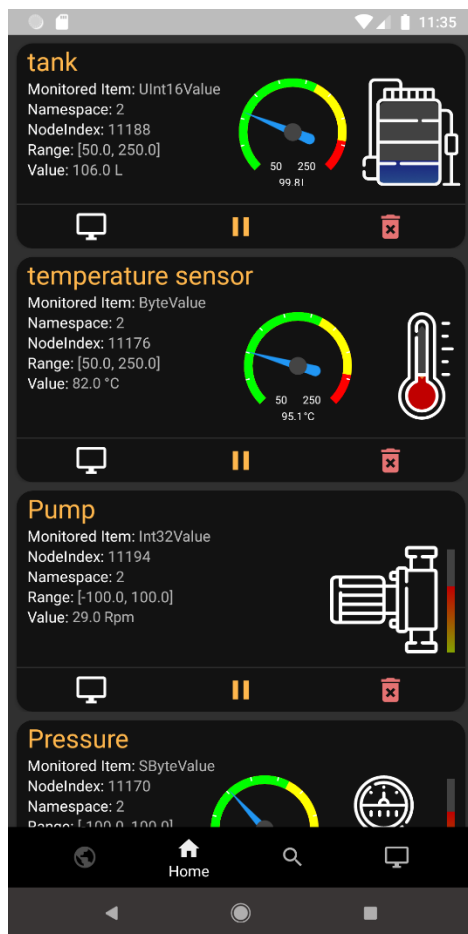


Figura 9:

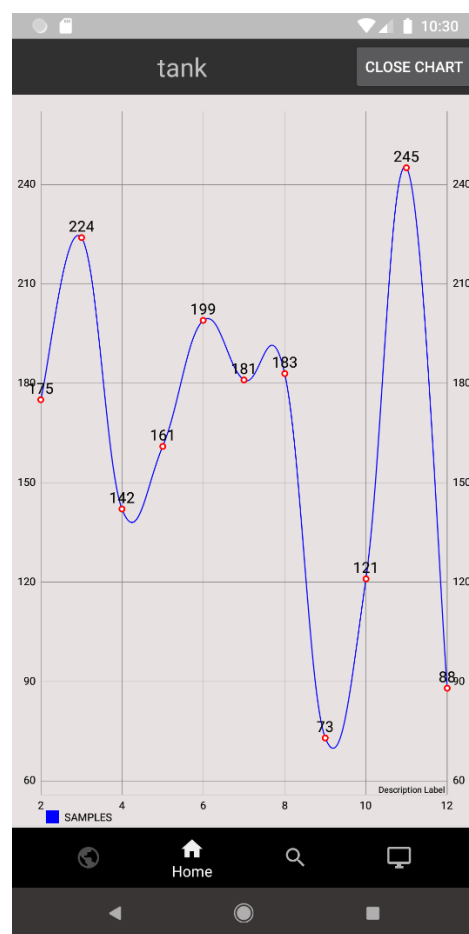


Figura 10: