

U.S. Forest Registry Client

User's Manual

1. Compiling and Starting the Application

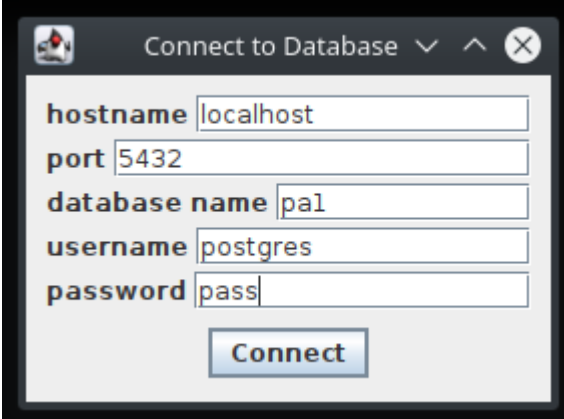
The entrypoint for the U.S. Forest Registry Client is the View class. Compile the application from outside the USForestRegistry directory, with View.java as the argument:

```
javac USForestRegistry/View.java
```

Then, invoke the application in a similar way, but with the JDBC PostgreSQL driver included in the classpath. For example:

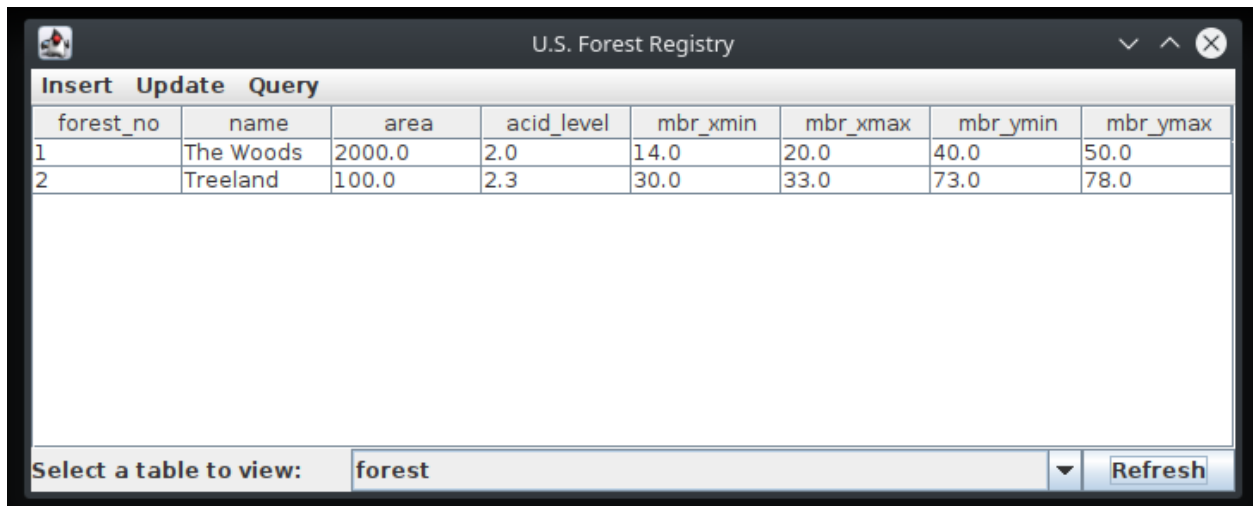
```
java -classpath lib/postgresql-42.3.3.jar:./ USForestRegistry.View
```

2. Connecting to the Database

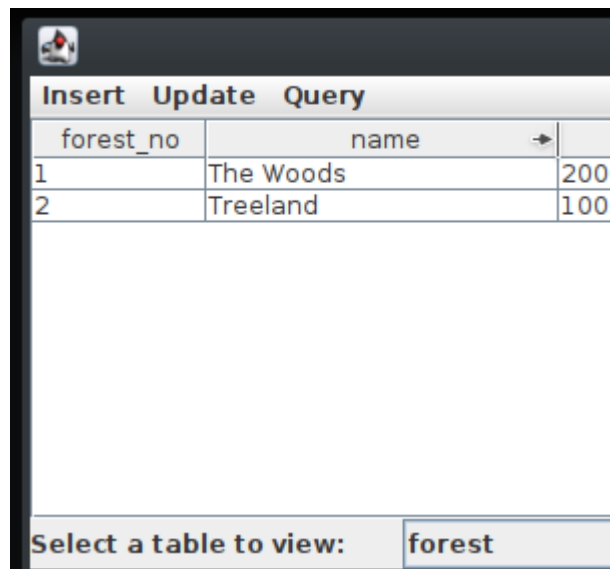
A screenshot of a Java Swing dialog box titled "Connect to Database". The dialog has a standard title bar with a minimize button, a maximize button, and a close button. Inside the dialog, there are five text input fields arranged vertically. The first field is labeled "hostname" and contains the text "localhost". The second field is labeled "port" and contains the text "5432". The third field is labeled "database name" and contains the text "pal". The fourth field is labeled "username" and contains the text "postgres". The fifth field is labeled "password" and contains the text "pass". Below these fields is a single button labeled "Connect".

After starting the application, you will be greeted with a database connection/login screen. Enter your credentials and hit "Connect", and then "OK" on the confirmation dialog, to connect to the database server.

3. The Main Interface

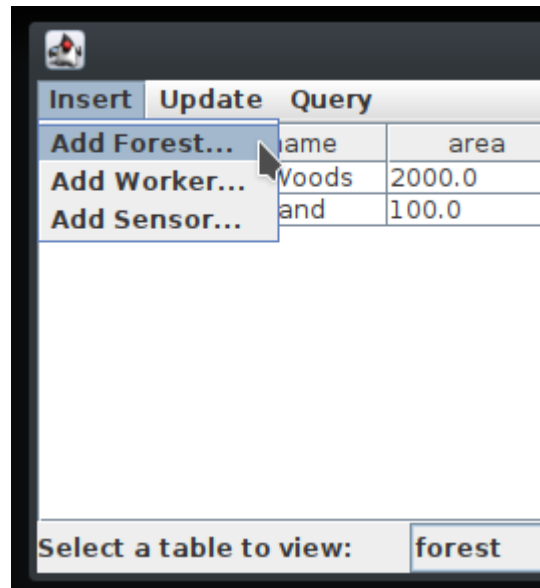


After logging in, you will see the main interface, pictured above. The Insert, Update, and Query menus at the top allow you to select one of the eight tasks that the application can perform. The table view in the center displays data from one of the eight tables in the underlying database. The drop-down menu at the bottom of the window allows you to select which database table to display, and the Refresh button at the bottom right reloads the currently displayed table, to reflect any new changes.



You can adjust the width of the displayed table columns by clicking and dragging on the column borders.

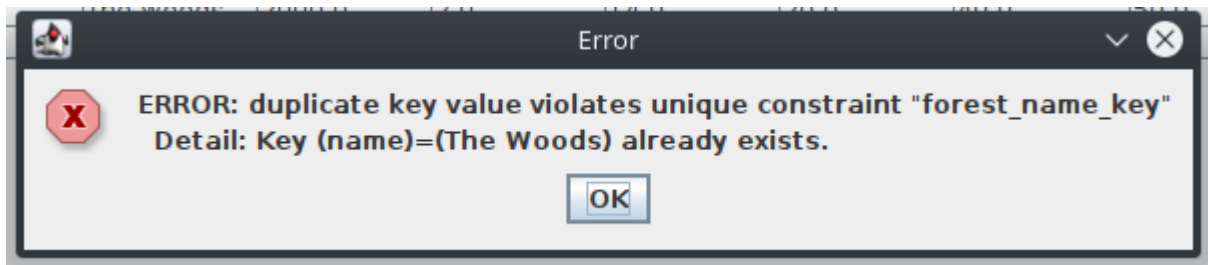
4. Performing a Task



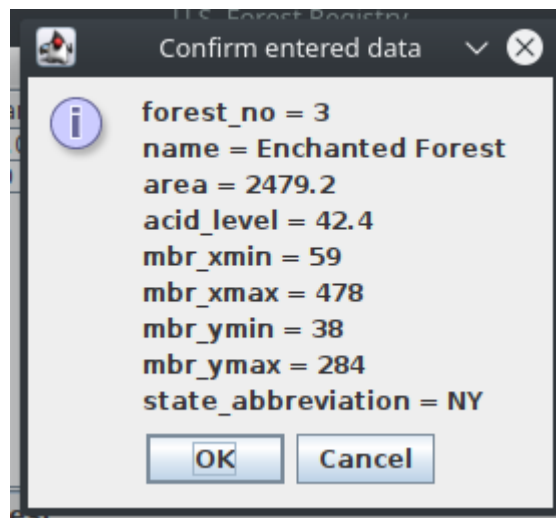
The process of adding a new forest to the database is shown here as an example of performing a task. Begin by selecting the “Add Forest...” option from the Insert menu.

The screenshot shows a dialog box titled 'Add Forest'. It has a title bar with a close button. The dialog contains several input fields with the following values: 'forest_no' is 3, 'name' is 'The Woods', 'area' is 2479.2, 'acid_level' is 42.4, 'mbr_xmin' is 59, 'mbr_xmax' is 478, 'mbr_ymin' is 38, 'mbr_ymax' is 284, and 'state_abbreviation' is 'NY'. At the bottom of the dialog, there are two buttons: 'Insert' and 'Cancel'.

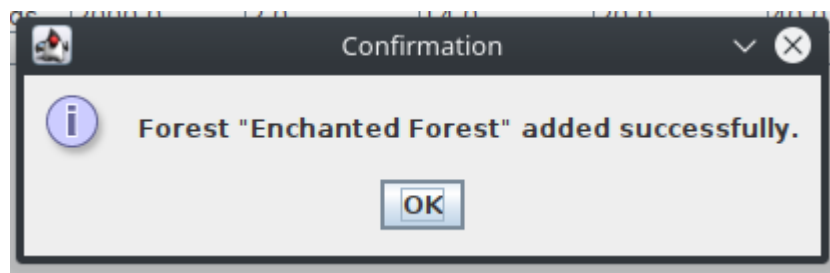
You will see a dialog box pop up, asking for user input for a number of fields. You can fill them in without worrying about data types: varchar types don't require quotes, and if you enter a decimal number in a field that only accepts whole integers, the field will automatically remove the decimal portion of the number when you click on another field or the Insert button. (The one data type that requires a specific format is the timestamp type, discussed later). When you're done, you can press the Insert button.



If the data you've entered causes a database error, you will see an Error dialog box pop up. In this case, we tried to insert a new forest with the same name as an existing forest, which is not allowed. Click the OK button or the "X" button to go back and edit your data.



Once you've entered valid data and you've clicked the Insert button, you will see a confirmation dialog appear, giving you a chance to review what you've entered before you make your changes final. At this point, you can click the Cancel button or the "X" button to go back and edit your data further, or, you can click the OK button to commit your changes.



A Confirmation dialog will appear, showing you a task-specific confirmation of a successful commit. (If instead we were performing one of the two Query tasks, we would get back a dialog box showing a readout of data from the database, instead of a confirmation message).

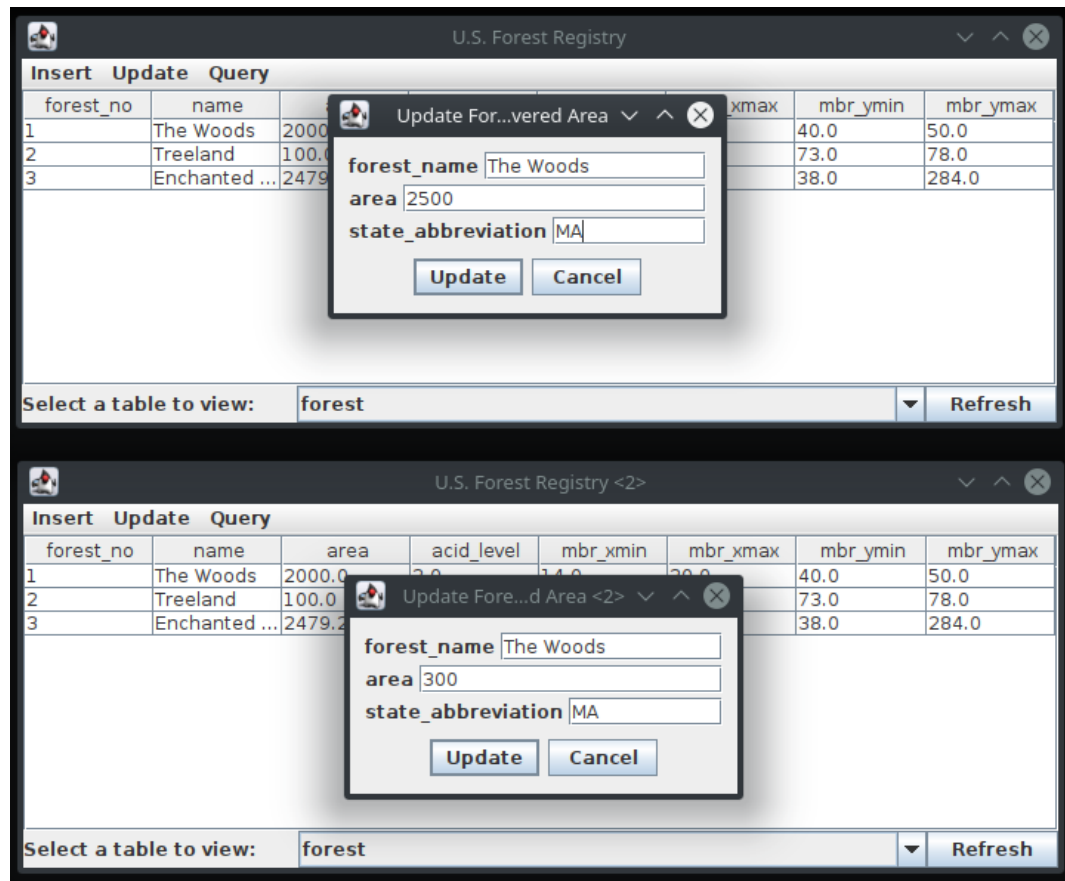
forest_no	name	area	acid_level	mbr_xmin	mbr_xmax	mbr_ymin	mbr_ymax
1	The Woods	2000.0	2.0	14.0	20.0	40.0	50.0
2	Treeland	100.0	2.3	30.0	33.0	73.0	78.0
3	Enchanted Forest	2479.2	42.4	59.0	478.0	38.0	284.0

Select a table to view: forest Refresh

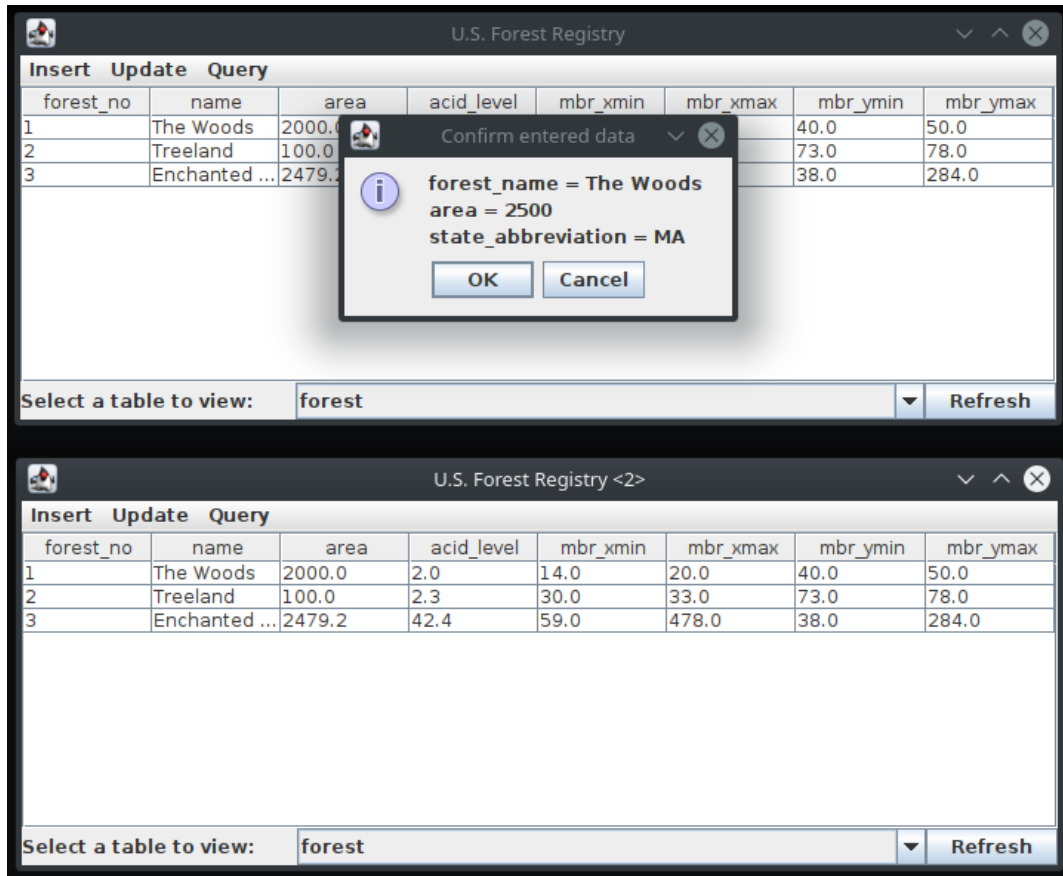
Back in the main interface, after refreshing the view of the forest table, our newly added forest is visible. If this newly entered forest was located in a state that was not yet present in the database's state table, the new state's abbreviation will also now be visible in the view of the state table.

5. Concurrent Access

The U.S. Forest Registry Client supports multiple instances of itself being run in parallel, connected to the same database. For the most part, it operates the same as if only one instance is running, with the exception of serialized access errors. These errors occur whenever two or more instances of the Client attempt to modify the same database rows at the same time. This can happen while performing any of the Update tasks, but it is shown below with the Update Forest Coverage task as an example. (An error is also raised in a similar way when two Insert tasks try to simultaneously create new rows that together would violate a UNIQUE constraint).

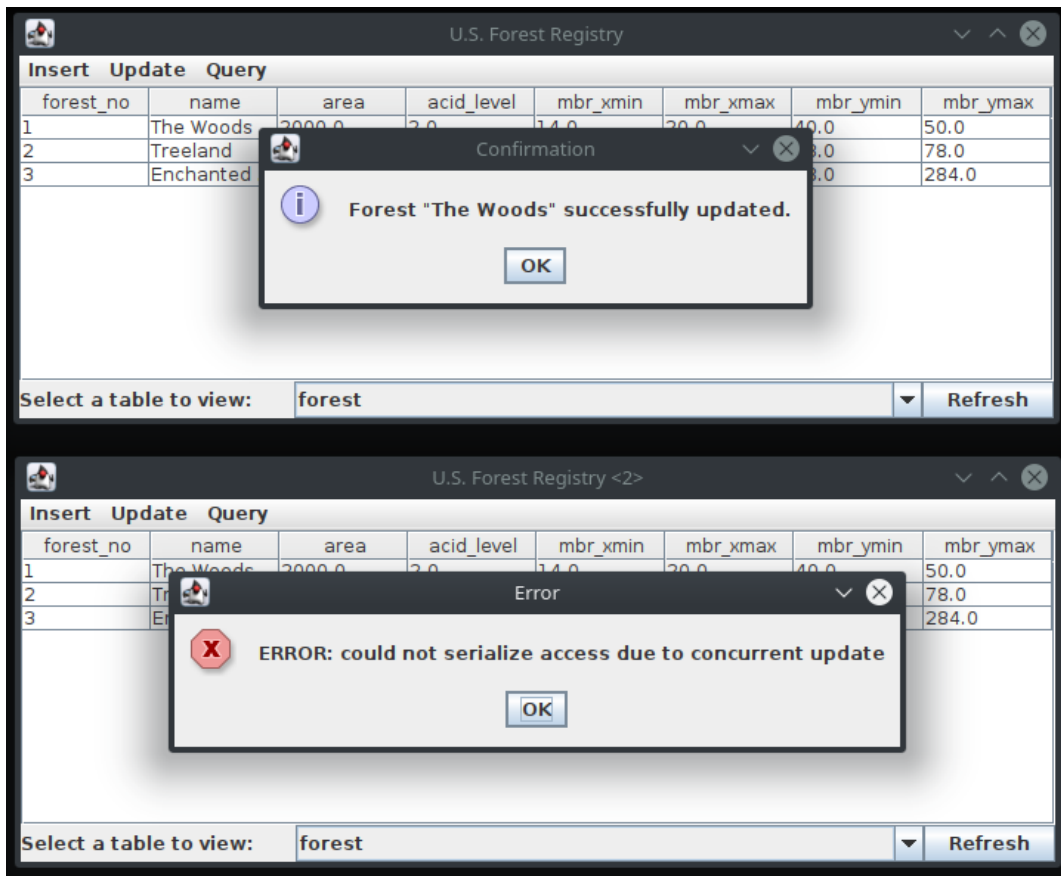


The top instance of the Client is getting ready to change The Wood's area to 2500, while the bottom instance is getting ready to change the area to 300. At this point, when the initial data entry dialog box is displayed, no indication has been given to the underlying database that any data is about to change. If the user of one Client did not click anything else past this point, the user of the other Client could make the change to The Wood's area like normal.



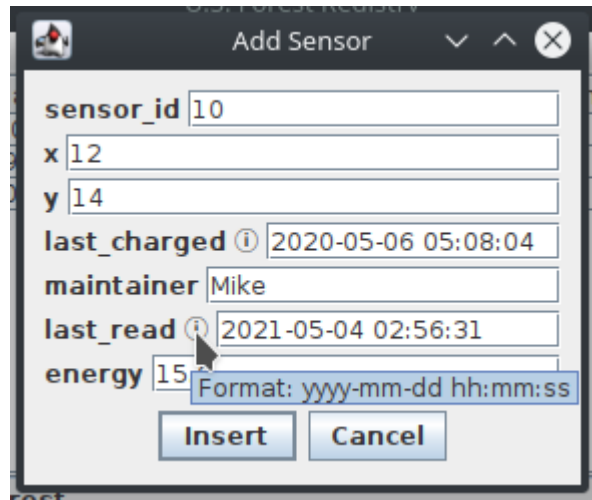
But now, both users have attempted to update the same row in the database by clicking the Update button (in this example, the top user clicked first, and then the bottom). The top Client shows the usual data confirmation dialog, but the bottom Client pauses and shows nothing while waiting for the top user to complete the action. This prevents inconsistent state from occurring as a result of concurrent data modification.

If the top user clicks Cancel, she gets the chance to go back and edit her data before making her changes final, meanwhile the bottom user will be presented with his own data confirmation dialog. But if the top user clicks OK, her changes will be successfully committed, and the bottom user's update attempt will fail with a serialized access error.



Here, the top user has clicked OK, and received a confirmation message indicating success. The bottom user's changes were not committed and the error is shown. When the bottom user clicks OK, he can change his entered data and attempt to update the database again.

6. Miscellaneous points

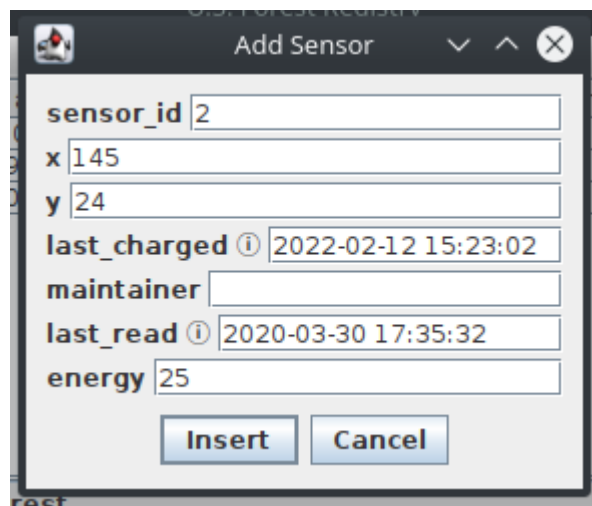


The screenshot shows the 'Add Sensor' dialog box with the following fields and values:

Field	Value
sensor_id	10
x	12
y	14
last_charged	2020-05-06 05:08:04
maintainer	Mike
last_read	2021-05-04 02:56:31
energy	15

A tooltip is displayed over the 'last_read' field, showing the format: `Format: yyyy-mm-dd hh:mm:ss`. The dialog has 'Insert' and 'Cancel' buttons at the bottom.

As mentioned earlier, timestamp data types are the only fields that require a special format. The format is “yyyy-mm-dd hh:mm:ss”, without the quotes. If you need a reminder, all timestamp fields have a small “i” icon next to them that you can mouse over, and a tip will appear to tell you about the format.



The screenshot shows the 'Add Sensor' dialog box with the following fields and values:

Field	Value
sensor_id	2
x	145
y	24
last_charged	2022-02-12 15:23:02
maintainer	
last_read	2020-03-30 17:35:32
energy	25

The dialog has 'Insert' and 'Cancel' buttons at the bottom.

In order to add a new sensor with no maintainer, simply leave the maintainer field blank, and it will be interpreted as a SQL null value. This null value will be reflected in the entered data confirmation dialog.

7. Possibilities for Improvements

The U.S. Forest Registry Client could be improved by using a proper password field on the login screen, which prints dots as the user types, instead of reflecting the user's password in plaintext.

Additionally, if the application were to be extended to have more general functionality, it would be helpful to allow non-varchar input fields to be submitted with null values – right now, only varchar fields like sensor.maintainer can be submitted with null values. If the user leaves other types of fields blank and clicks the submit button, the user will see a popup reminding them to fill in a value for those fields.

In the same vein, a more generally capable application would also have a way to distinguish between varchar fields submitted with empty strings, and those submitted with null values. As stated, empty varchar fields are currently interpreted as null values only.