Argus

PROJECT DEVELOPER GUIDELineS  
**09-09-2020**

# Overview

## Project Background and Description

|  |  |
| --- | --- |
|  | Hardware & Tools Used  1. Operating system Ubuntu Linux 16.04.6  2. Apache version 2.4.43  3. PHP version 7.2 (Laravel 6)  4. MySQL version 5.7.30-0 ubuntu0.16.04.1  5. Mosquito  6. Python 3.5 |

## Project Scope

|  |  |
| --- | --- |
|  | Mosquito data inserting to the Mysql dbo\_payloader table through Python Script. There are several Cron jobs running in specified intervals to process the data as per the project requirement. |

## Database Structure and Tables.

|  |  |
| --- | --- |
|  | Following are the tables used.   * **algorithm-** *To store algorithm of various agents.* * **algorithm\_sensor -** *To store sensor, hub, conditions of various algorithm of various agents.* * **chart -** *To store the unit and type of chart varies on the basis of unit in application.* * **dbo\_payloader-** *Mosquito data storing in this table.* * [**dbo\_payloaderalgorithmtemp298**](https://mqtt.eurozapp.eu/zapadmdb/sql.php?server=1&db=mqttdata&table=dbo_payloaderalgorithmtemp298&pos=0)*- 298 is the agentid or userid,runtime table created for each agent for storing data from dbo\_payloader for sending push notifications using 1 condition algorithm(none).This table created at the time of adding agents and deleted while removing an agent.* * **dbo\_payloaderalgorithmtempandor298-** *298 is the agentid or userid,runtime table created for each agent for storing data from dbo\_payloader for sending push notifications using more than 1 condition algorithm(and,or,and-and,or-or,and-or,or-and etc).This table created at the time of adding agents and deleted while removing an agent.* * **dbo\_payloadercharttemp-** *To store sensor chart details temporarily by logged user for generating charts.* * **gateway\_groups-** *To store gateway groups* * **loc-** *To store location template* * **log\_details -** *To track login and logout of various users* * **roles-** *To store type of users(1-Admin,2-Agent-Only 2 users)* * **role\_user –** *To mapping users with their roles* * **sensors-** *To store the sensors of various agents* * **sensor\_groups-** *Master for various sensor groups.* * **sensor\_hubs-** *To store hubs of various agents* * **settings-** *Template for sending emails* * **types-** *To store sensor types* * **userdatamessagesagent-** *To store messages of various agents(pushnotification messages)* * **users-** *to store user details(Admin,Agent)* * **weather-** *Mapping agent with location* * **sensordata(master table)-** *For populating in sensor dropdowns while assigning new sensors to agents.* * **hubdata(master table)-** for populating in hub dropdowns while assigning new hubs to agents. |
|  |  |

## Source Code Details

|  |  |
| --- | --- |
|  | Framework used is laravel6 with model–view–controller architectural pattern (MVC). We have two Modules - I. **Admin** and II. **Agent**. Admin -Path(argus/app/Http/Controllers/Admin) Following are the controllers and files in Admin module   * **AdminsController -** Managing Users and sending mails * **AlgorithmController -** Algorithm handling and Hub Sensor Report(Table,Chart) of Admin * **Agents Controller-** For Handling Agents * **Emails Controller-** Sending Mail * **GatewayGroup Controller-** For Handling Gateway Groups * **Home Controller-** Code for admin dashboard,weather details,agent-login,logout time etc * **Report Controller-** Code for Sensor Time Table,Chart,Push Notification etc * **Sensor Controller-** Code for handling sensors of various agents * **Sensor hub Controller-** Code for handling sensor hubs of various agents * **Settings Controller-** Code for settings module(Unit,chart for various units,sensor type etc) * **Web.php-** used for routing   Path(argus/routes/web.php)   * **Blade files**   Path(argus/resources/views/admin)   * **.env**   database settings (inside argus folder).   * Exporting Excel-Laravel Excel Export used(argus/app/Exports)   ------------------------------------------------------------------------------------ Agent -Path(argus/app/Http/Controllers/Agent) Agent Module following are the controllers and files used.   * **AlgorithmController-** Algorithm handling and Hub Sensor Report(Table,Chart) of Agents * **Agents Controller-** For Handling Agents * **GatewayController-** For Handling Gateway Groups(agent) * **GatewaygroupController-** For Handling Gateway Groups(agent) * **Home Controller-** Code for agent dashboard, * **Profile Controller-** Profile for agent * **Report Controller-** Sensor Time Report(Table,Chart),Push Notification * **SensorhubController-** Sensor Hubs of Agent * **Settings Controller-** Code for settings module(Unit,chart for various units,sensor type etc) * **UserController-** For Managing Users * **Blade files**   Path(argus/resources/views/agent)   * **Exporting Excel**- Laravel Excel Export used(argus/app/Exports) |

## Points to note.

|  |  |
| --- | --- |
|  | * **Admin and Agent Sensor Time Report:** we are using temporary table to store the data for the searched criteria (User) for displaying sensor chart report.   **Function** used is savecharttempdata in ReportController.  Table used is **dbo\_payloadercharttemp** with **loginid**(agent or admin logged into the application)   * **Login Customization done inside below path :** argus/vendor/laravel/framework/src/Illuminate/Foundation/Auth/AuthenticateUsers.php * **Algorithm-Single condition logic:**   We are using Cron job for push messages in the **userdatamessagesagent** table for various agents.  We used two cron jobs, one for **single condition** and the second **for more than 1 condition**. For push notification, we are taking data from **dbo\_payloader** and stored it in each agent temporary table(eg: **dbo\_payloaderalgorithmtemp298**).298-agentid.  From this table **dbo\_payloaderalgorithmtemp298**-program will take data for processing.  Cron job, for fetching data from **dbo\_payloader** to **dbo\_payloaderalgorithmtemp298** is **pushsingleagent.php** and single condition algorithm program is **pushnotagent.php**  both of these files are in argus root folder.  **Pushnotagent.php(code details)**   * + Get sensors of agent from algorithm with single condition-more condition flag—0   + Get data from table where processedflag is 0. (**dbo\_payloaderalgorithmtemp298**)   + Compare its value with algorithm sensor value   + If it meets condition data or message inserted into **userdatamessagesagent** table.   + Once record is processed data deleted from this **dbo\_payloaderalgorithmtemp298** table   + Records which are not these agent it will be deleted.   + Cron job triggered for every 1 minute. * **Algorithm-More than 1 condition logic**   We are using Cron job for pushing messages in the **userdatamessagesagent** table of various agents.  We used two Cron job one for single condition and the second one for more than one condition. For push notification, we are taking data from **dbo\_payloader** and stored it in each agent temporary table (eg: **dbo\_payloaderalgorithmtempandor298**).298-agentid.  From this table **dbo\_payloaderalgorithmtempandor298**-program will take data for processing.  Cron job, for fetching data from **dbo\_payloader** to **dbo\_payloaderalgorithmtempandor298** is pushallagent.php and more than 1 condition algorithm program is pushnotmsgagent.php  Both of these files are in argus root folder.  **Pushnotmsgagent.php(code details)**   * Get sensors of agent from algorithm with more condition flag—1 * Get data from table where processedflagall is 0. **(dbo\_payloaderalgorithmtempandor298**). * Compare its value with algorithm sensor value * If it meets condition, data or message inserted into **userdatamessagesagent** table. * Once the record processed then the data deleted from this **dbo\_payloaderalgorithmtempandor298** table. * Also records which are not these agent also deleted. * Cron job triggered for every 1 minute. |
|  |  |

## Cron Jobs Used in Argus.

|  |  |
| --- | --- |
|  | * **Push Notification Single-**    + pushsingleagent.php   + pushnotagent.php * **Push Notification More than 1 Condition**   + pushallagent.php   + pushnotmsgagent.php * To reset the logout status of users:   + If users are idle or closed the application without clicking logout then Cron job reset loginstatus to 0 -sesexpnew.php.   + **sensor.php** - to fetch new sensors from **dbo\_payloader** and storing in **sensordata** (master table)-for populating in sensor dropdowns.   + **hub.php-** to fetch new hubs from **dbo\_payloader** and storing in **hubdata**(master table)-for populating in hub dropdowns |
|  |  |

## Python File

|  |  |
| --- | --- |
|  | * Python is using for Mosquito data to insert data into dbo\_payloader table.   **Python File**- mqtt-db.py |

## To Run Source Code

|  |  |
| --- | --- |
|  | Source code has itself developer guide lines.  **Configuration Instruction:**  1. Install Operating system Ubuntu Linux 16.04.6  2. Install Apache version 2.4.43  See the document : Install\_Apache.txt  3. Install PHP version 7.2  See the document : Install\_PHP7.2.txt  4. Install MySQL version 5.7.30-0 ubuntu0.16.04.6  See the document : Install\_MySQL5.7  5. Install Mosquito  See the document : Install\_Mosquitto.txt  6. Install Python 3.5  See the document : Install\_Python\_3.5.txt  7. Create mySql database ‘mqttdata’ and then import the database mqttdata.sql  8. Copy and Paste the source file in your destination. Add your new db user name and password, and URL in the .env file.  If application not working, then run the following command from your application root path  a. php artisan cache: clear;  b. php artisan config:clear;  If application is still not working then update vendor file, the command you have to use “composer update”.  \*If updating or installing vendor files—keep a backup of argus/vendor/laravel/framework/src/Illuminate/Foundation/Auth/AuthenticateUsers.php and replace it with new AuthenticateUsers.php |
|  |  |