piseduce_webui
commit: a78124eb05882533e2e
17 May 2021

DB table: user

- email
- password
- ssh_key
- confirmed_email
- is_authorized
- is_admin

The user table registers the user accounts that are identified by a unique email (primary key).

The is_admin boolean field must be equal to 1 for users with administration privileges.
These special users can access to the admin pages of the webUI (user management, node management, VPN and network, etc.).

From the user admin page, administrators must allow users to log in after their registration. This operation sets the is_authorized field to 1.

The confirmed_email field is equal to 1 if users are clicked on the confirm email link (Not implemented).

DB table: agent

- name
- type
- ip
- port
- status
- token

The agent table registers the agents used by the webUI to provide nodes.

Registered agents are identified from the name field. Names are unique (primary key).

The agent type is defined by the type of the nodes that it manages. This type can be raspberry, node or sensor.

The field ip is the ip of the agent.

The port is the port used by the agent (see the configuration file of the agent).

The status is equal to connected or disconnected. The webUI does not query disconnected agents.
Agents are flagged disconnected when the webUI try to query them without receiving any answer.
Agents must reply within 6 seconds otherwise there are also flagged disconnected.
Administrators can reconnect agents from the agent admin page.

To communicate with agents, the webUI must send the token defined in the agent configuration file.
The token represents the token agent.

piseduce_agent
commit: a8dcf3ec172bfad31a94
17 May 2021

DB table: rasp_node

- node_name
- prop_name
- prop_value

The node table registers the nodes managed by the agent.
Nodes have properties as IP, model, etc.

The prop_name field is the unique name of the properties.

The prop_value field is the value of the properties.

DB table: schedule

- node_name
- owner
- bin
- start_date
- end_date
- status
- action_state

The schedule table registers the node reservation of the users. There is one reservation per node.

The owner field is filled by the user email. Users can make one reservation per node
(the primary key is (node_name, owner).

The bin field is only used by the webUI to group nodes in the user interface.

The start_date is the date of the beginning of the deployment.

The end_date is the date on which the destroy process of the reservation starts.

The status of the node is one of the following values:
 - configuring: the node is reserved by the user but the properties required to deploy the node are not set.
 - ready: the node is properly configured. If the action_state field is empty, the agent waits until the start_date to start the deployment by executing the deploy action.
 - in_progress: one action on the node is executing, for example, the deploy action. Possible actions are: deploy, destroy, reboot.
    When the action is completed, the node status becomes ready again.

The action_state field is used to indicate the current state of ongoing actions. These states are defined in the directory associated to the node type.
For example, the states of the raspberry nodes are defined in raspberry/states.py.

DB Table: action_prop

- node_name
- prop_name
- prop_value
- owner

The action_prop table registers the properties required to execute actions as the deploy action.
These properties are provided by users.
When all action properties are defined, the node status changes from configuring to ready.

The prop_name field is the unique name of the properties.

The prop_value field is the value of the properties.

The owner is the user email. Users can make only one reservation per node.

DB Table: action

- node_name
- node_ip
- environment
- process
- state
- state_idx
- updated_at

The action table registers the ongoing actions. Three types of action can be executed: deploy, reboot, destroy.
The action can be added by the agent or by users via the webUI.

The node_name is the name of the node.

The node_ip is the IP of the node.

The environment is the name of the environment to deploy on the node.
The list of the available environments is defined by the environment table.

The process is the type of the action (deploy, reboot, destroy). The process defines the sequence of states that are executed to configure the node.
This sequence of states highly depends on the node type. The list of these states are located in the node type directory.
For example, for raspberry nodes, the states are defined in raspberry/states.py.

The state field is the name of the current state of the action.
During the action execution, the action goes through different states that depends of the node type (raspberry, sensor, server).
The last state of actions is one of the following: deployed, lost, rebooted, destroyed.

As one state can be executed many times, we used the state_idx field to know exactly the next state to execute.

The field updated_at is the date on which the last change of the action state occurred. This date allows to detect if the action is stalled.
State timeouts are defined in the states.py file. If timeouts are reached, a reboot can be executed or the action state can change to lost.
In the latter case, the action execution is stopped and the node can not be used by the user.