Dashboards

Main Project Controller

Your report must have a PHP class and file with the same name in the Controllers folder. The namespace must be set correctly and the class should extend the Report class like in our example controller below. The class should have a constructor where parent::__construct(); is called and must also have
declared activate, deactivate and uninstall functions that must return true. The constructor is also where you register your dashboards, which will be described in more detail below.

Controllers/TicketBacklog.php

```
<?php declare(strict_types=1);namespace AddonsPlugins
TicketBacklogControllers;use AddonsReportsTicketBacklogReports
ExampleDashboard;use AppModulesReportAddonReport;class
 TicketBacklog extends Report{    public function __construct()    {
        parent::__construct();        $this->
registerReportDashboards([ExampleDashboard::class]);    }
    /**     * @return bool     */    public function activate()    {
        return true;    }    /**     * @return bool     */    public function
 deactivate()    {        return true;    }
    /**     * @return bool     */    public function uninstall()    {
        return true;    }}
```

Usually no extra logic is required while activating, deactivating or uninstalling a report. However, if you need to for example add a new datatable table, you can follow the documentation from our

plugin development guide as it is part of the same add-on system.

Your dashboards should be placed at the top level of
the Reports folder and extend
the AppModulesReportAddonDashboardsDashboard class, like below.

Reports/Overview.php

```
<?php declare(strict_types=1);namespace AddonsReports
TicketBacklogReports;use AddonsReportsTicketBacklogReportsCards
OverviewBacklogNumber;use AppModulesReportAddon
DashboardsCardCollection;use AppModulesReportAddon
DashboardsDashboard;use AppModulesReportAddonDashboards
FilteringFilter;use AppModulesReportAddonDashboardsFiltering
Timeframe;class Overview extends Dashboard{    public
 Timeframe $timeframe = Timeframe::DISABLE;    public Filter
 $filter = Filter::NONE;    public function name(): string    {
        return 'Overview';    }    public function description(): string    {
        return
 'An overview of tickets in the backlog
.';    }    public function loadCards(CardCollection $collection): void    {
        $collection->add(new BacklogNumber);    }}
```

The $timeframe property can be set to Timeframe::ENABLE if your report should have timeframe options, so you can limit it to run on say the last week or 30 days.

```
public Timeframe $timeframe = Timeframe::ENABLE;
```

The $filter property can be set to Filter::TICKET or Filter::USER if your report should have filtering options. These only apply if the cards data are on the Ticket or User models which will be covered later.

```
public Filter $filter = Filter::TICKET
```

The name and description methods should have a string output. You can use [redacted] if the system will be used in multiple languages.

```
public function name(): string{     return trans(
'Reports#TicketBacklog::lang.overview');}
```

Finally the report cards should be loaded in the loadCards method. The cards should be added in order that you wish to show them. The next section will cover writing cards.

```
public function loadCards(CardCollection $collection): void{
   $collection->add(new BacklogNumber);    $collection->add(new
 BacklogByStatusGraph);}
```

Registering Dashboards

All dashboards must be registered in the main report controller constructor using the registerReportDashboards method. Multiple dashboards can be set in the array and they will display in this order in the reports sidebar.

```
public function __construct(){    parent::__construct();    $this->
registerReportDashboards([        AddonsReportsTicketBacklogReports
Overview::class,        \ Other dashboards here    ]);}
```

Online URL: https://docs.supportpro.vn/article/dashboards-237.html