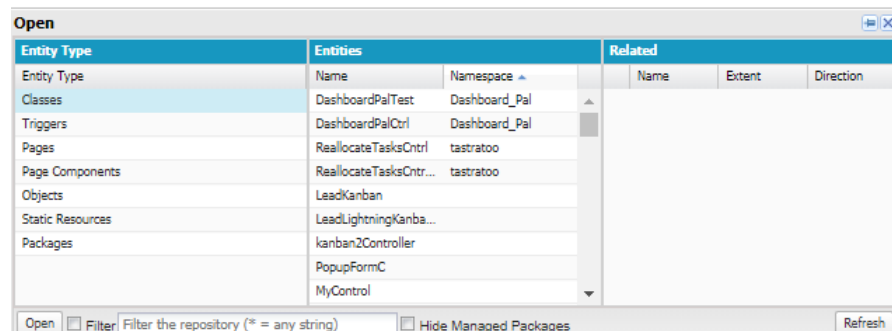


```
onLoad: function(component, event) {
    console.log('onLoad call');
    //call apex class method
    var action = component.get('c.fetchProject');
    action.setCallback(this, function(response) {
        //store state of response
        var state = response.getState();
        if (state === "SUCCESS") {
            //set response value in ListOfContact attribute on component.
            component.set('v.ListOfContact', response.getReturnValue());
            // set default count and select all checkbox value to false on load
            component.set("v.selectedCount", 0);
            component.find("box3").set("v.value", false);
        }
    });
    $A.enqueueAction(action);
},
```

Figure IV.9 – Interface de Contrôleur «ListProjetHelper.Js»

4.2 Implémentation du back end

Afin d'avoir une partie back end interopérable, nous avons implémenté des classes Apex. La figure IV.10 présente la liste des classes Apex de notre application.



Entity Type	Entities	Related
Entity Type	Name	Namespace
Classes	DashboardPalTest	Dashboard_Pal
Triggers	DashboardPalCtrl	Dashboard_Pal
Pages	ReallocateTasksCtrl	tastratoo
Page Components	ReallocateTasksCtrl	tastratoo
Objects	LeadKanban	
Static Resources	LeadLightningKanba...	
Packages	kanban2Controller	
	PopupFormC	
	MyControl	

Figure IV.10 – Interface de la liste des classes Apex

* Contrôleur Apex «ListProjetController.apxc»

La figure IV.11 explique l'architecture de notre composant Lightning qui se lie aux méthodes d'action Apex sur la plateforme Salesforce.