WEB PROGRAMMING AND TESTING WITH A MODERN FRAMEWORK:ANGULARJS

Project Name:

Task Manager

Author:

Sabrina Micale

O55000259

sabrinamicale@michiweb.it

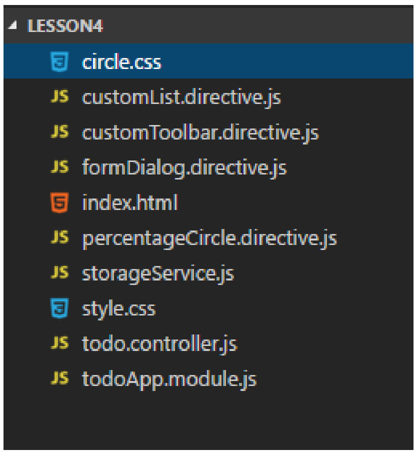
1. Features and modifications

The tasks have been extended to have more attributes such as tags, estimated work hours, subtasks and completion percentage.

The features added to the starting example are:

- Multiple selection for modifying status, priority and for deleting tasks.
- Search functionalities.
- Sorting of tasks by title, date, priority and completion in ascending and descending order.
- Modified creation interface to include all the tasks' attributes.
- Update interface to change task's title, description, date, ecc...
- Completion percentage circle to show the task's completion state based on how many of its subtasks have been completed.

2. Project's Architecture



```
LESSON4
    circle.css
    customList.directive.js
    customToolbar.directive.js
    formDialog.directive.js
    index.html
    percentageCircle.directive.js
    storageService.js
    style.css
    todo.controller.js
    todoApp.module.js
```

The files added to the starting example are:

- customToolbar.directive.js, containing the directive for the action toolbar of the app.
- formDialog.directive.js, containing the directive for the content of the creation and update interfaces.

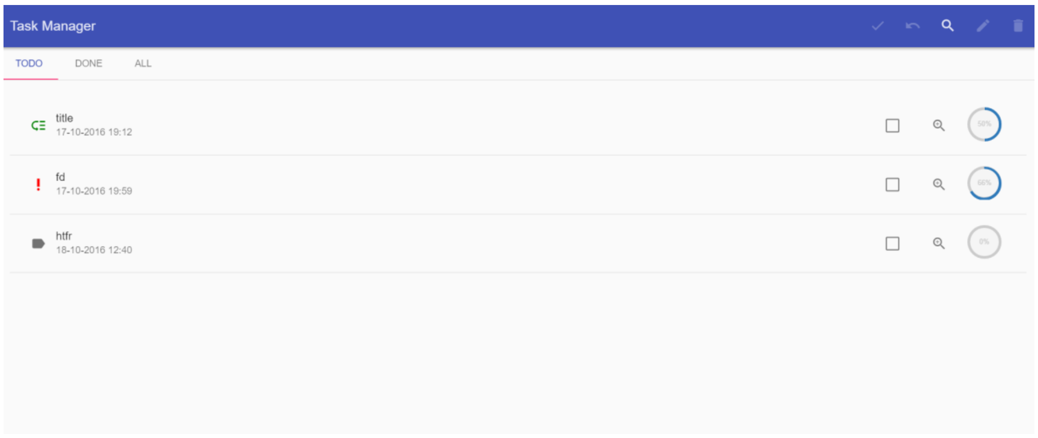
- percentageCircle.directive.js and circle.css, containing the directive and the css classes specifications to show the task's completion percentage.

Moreover the module and controller's definitions have been split in the files todoApp.module.js and todo.controller.js, respectively.

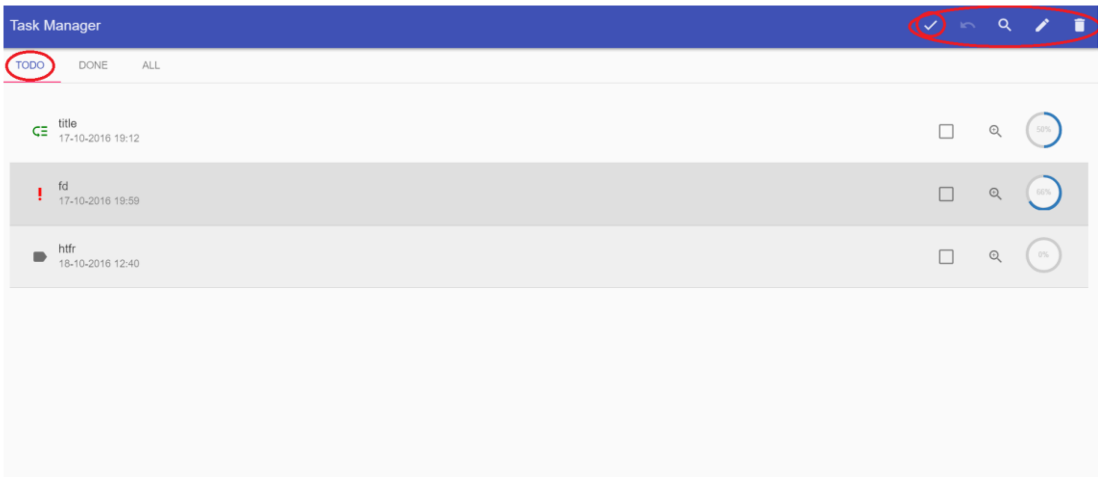
3. Features description

3.1 Multiple selection

The original project let the user select only a task at the time by clicking on the single task on the list, and then it is possible to delete the task by clicking on the delete button on the toolbar.
In the final app it is possible to select multiple tasks by clicking on them and it is possible to change the status and priority and to delete the selected items through the designed buttons in the toolbar. To achieve the multiple selection, every task has a Boolean attribute 'selected' set to true if the task has been selected, while the original variable in the customList directive selectedItem is used to count the tasks selected.
Through the use of this variable, when no items are selected the buttons in the toolbar are disabled:



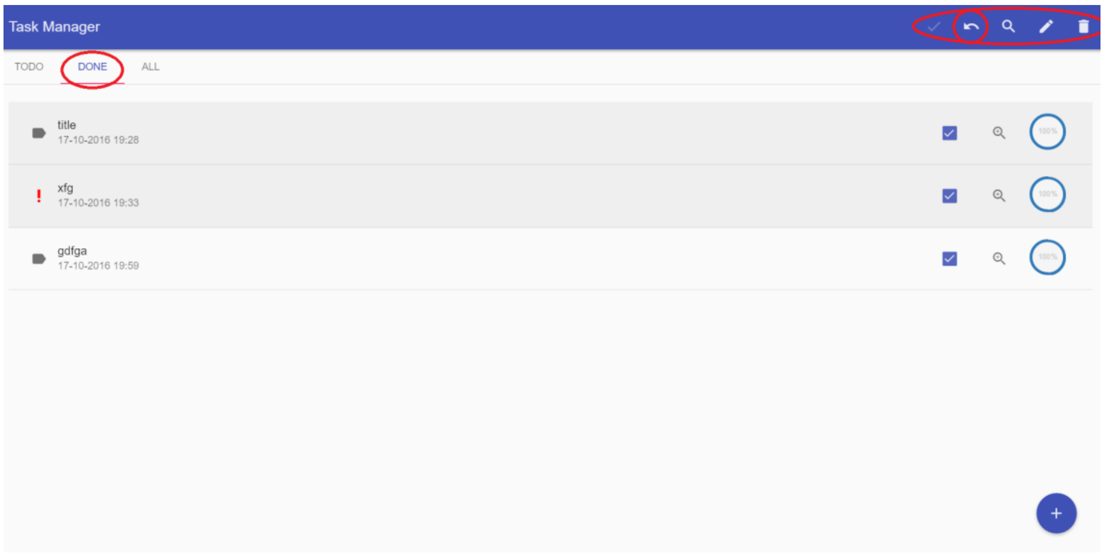
When some items are selected in the 'TODO' tab, the buttons to 'set to done', to change priority and to delete the selected tasks are active, while when the active tab is the 'DONE' one, the button to 'set the tasks to not done' is active in place of the 'set to done' one. They are both active when the tab selected is 'ALL'.



title
17-10-2016 19:12

fd
17-10-2016 19:59

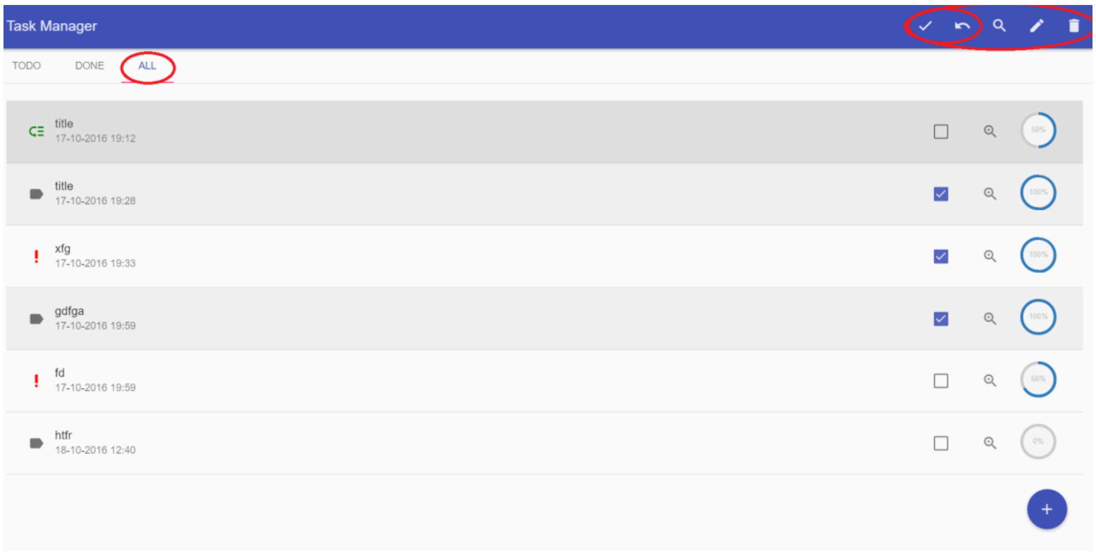
htfr
18-10-2016 12:40



title
17-10-2016 19:28

xfg
17-10-2016 19:33

gdfga
17-10-2016 19:59



title
17-10-2016 19:12

title
17-10-2016 19:28

xfg
17-10-2016 19:33

gdfga
17-10-2016 19:59

fd
17-10-2016 19:59

htfr
18-10-2016 12:40

```
'<md-button ng-disabled="customToolbarCtrl.selectedItem ==0 ||
customToolbarCtrl.disableFunction==customToolbarCtrl.disableApplyCondition" ng-click="customToolbarCtrl.applyFunction()"
class="md-icon-button" aria-label="Done">'+
                 '<md-tooltip md-autohide=\'true\'>Set to done</md-tooltip>'+
                 '<md-icon> done </md-icon>'+
              '</md-button>'+
              '<md-button ng-disabled="customToolbarCtrl.selectedItem ==0 ||
customToolbarCtrl.disableFunction==customToolbarCtrl.disableCancelCondition" ng-click="customToolbarCtrl.cancelFunction()"
class="md-icon-button" aria-label="Not Done">'+
                 '<md-tooltip md-autohide=\'true\'>Set to not done</md-tooltip>'+
                 '<md-icon> undo </md-icon>'+
              '</md-button>'+
...
```

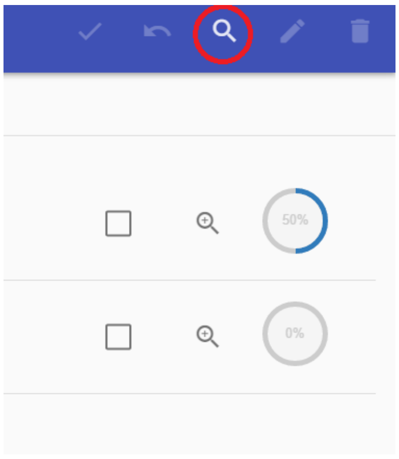
The buttons of the toolbar are collected in the directive customToolbar. The items, the count of selected tasks, as well as the functions called by the buttons and to disable/unable the buttons, are attributes passed to the directive:

```
  function directive() {
      return {
          scope: {},
          bindToController: {
              items: '=',
              selectedItem: '=',
              disableFunction: '=',
              applyFunction: '=',
              cancelFunction: '=',
              disableApplyCondition: "@",
              disableCancelCondition: "@",
              priorityFunction: '=',
              deleteFunction: '=',
                  ...
```

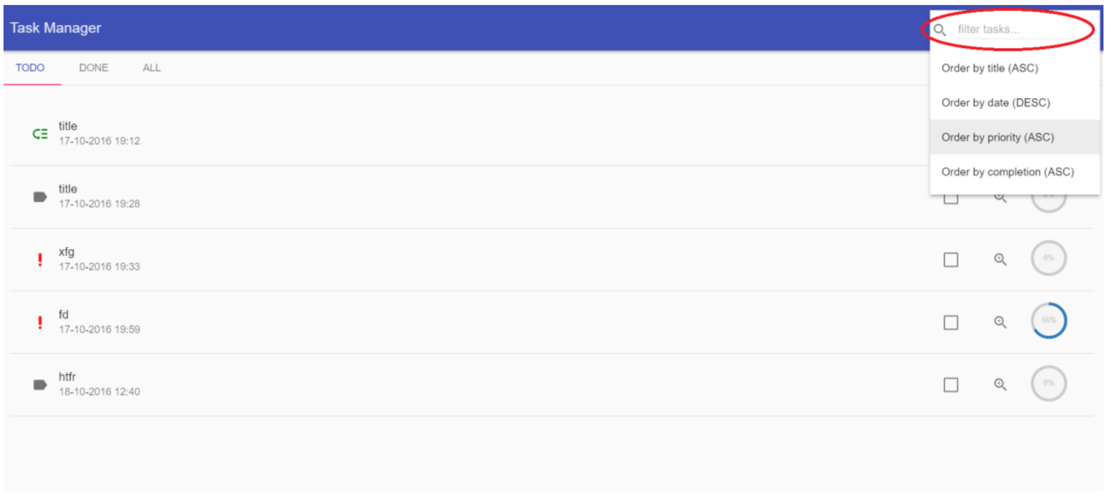
The tasks are automatically deselected when the user passes from a tab to another, or after an action is executed.
Like in the original project, it is still possible to change the priority or the status of a single task through the designed buttons showed with each task.

3.2 Search functionalities

The filter button is part of the toolbar and it is always active.



When the user clicks on it a menu is opened and it is possible to search tasks by writing in the input box 'filter tasks' which is the first item of the menu.



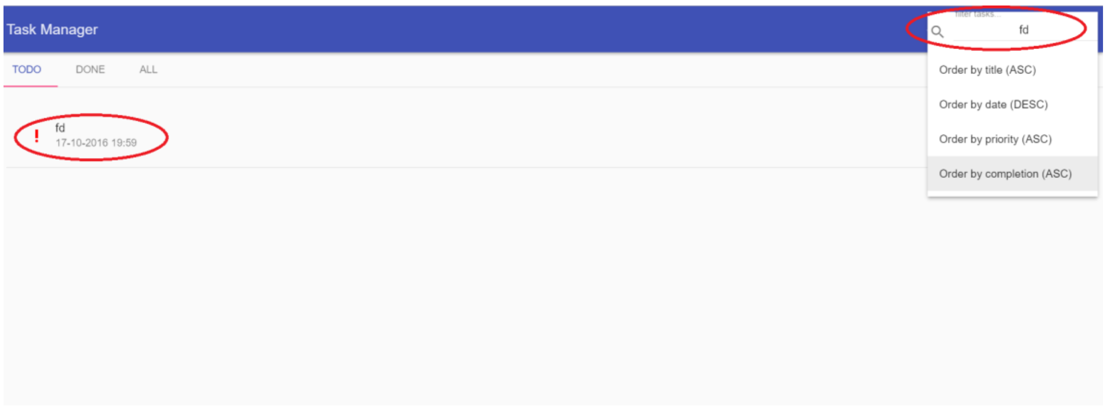
title
17-10-2016 19:12

title
17-10-2016 19:28

xfg
17-10-2016 19:33

fd
17-10-2016 19:59

htfr
18-10-2016 12:40



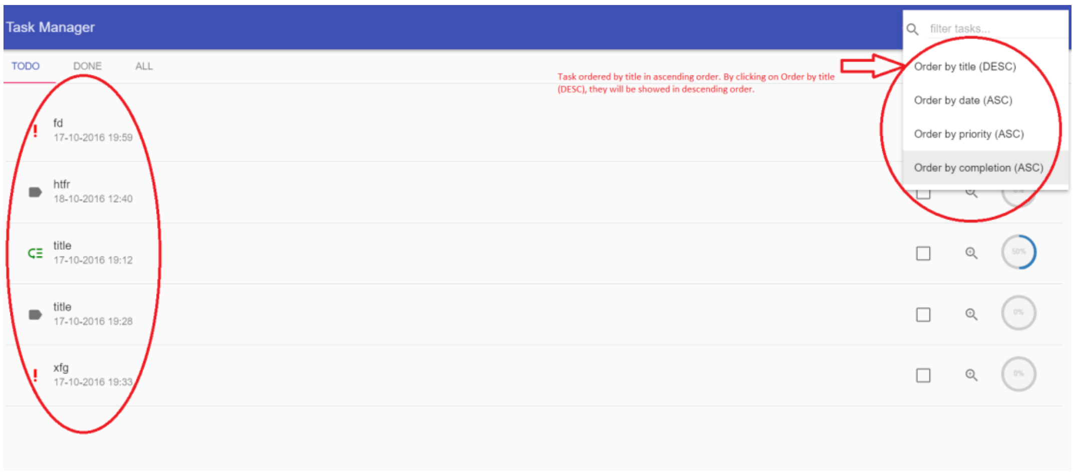
fd
17-10-2016 19:59

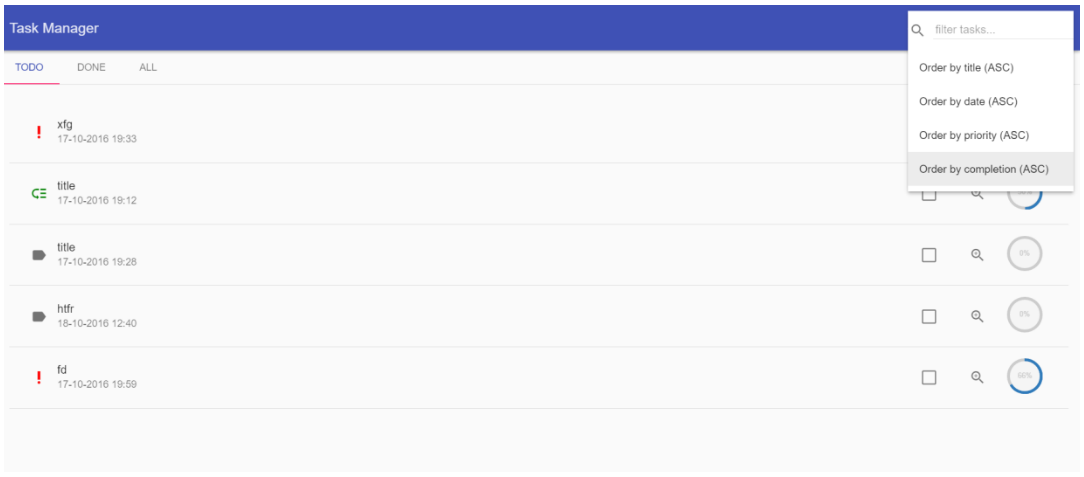
The tasks displayed are the ones with a title, description, tag or subtask's name that contains the written string.
The tasks are 'filtered' also in the other tabs till the user deletes the string in the input search box.
This is achieved through the use of a filter and the attribute filteredItems which is passed to both the customList and customToolbar directive.

```
<md-list-item class="md-2-line" ng-repeat="item in customListCtrl.items | filter: customListCtrl.filteredItems | filter: customListCtrl.filterFunction | orderBy: customListCtrl.orderBy" ng-class="item.selected == true ? \'selected\':\'\'" ng-click="customListCtrl.toggleSelection(item)">'
```

3.3 Task sorting

By default the tasks are sorted by title in ascending order.
Through the menu opened by the filter button, though, the user can sort the tasks also by date, priority and completion percentage in both ascending and descending order (the latter by clicking again on the same menu item after having sorted the tasks in ascending order).


Task ordered by title in ascending order. By clicking on Order by title (DESC), they will be showed in descending order.



The tasks sorting is achieved through the use of the ‘orderBy’ filter.

```
<md-list-item class="md-2-line" ng-repeat="item in customListCtrl.items | filter: customListCtrl.filteredItems | filter: customListCtrl.filterFunction | orderBy: customListCtrl.orderBy" ng-class="item.selected == true ? \'selected\':\'\'" ng-click="customListCtrl.toggleSelection(item)">'
```

Both the customList and customToolbar directives have an ‘orderBy’ attribute to let the user choose the sorting order.
In the customList directive the filter is applied, while the variable ‘orderBy’ is changed in the customToolbar directive through the function ‘orderItemsBy’ called when the user clicks on one of the sorting menu options.

```
function CustomToolbarController() {
   var vm = this;
   //Order items
   vm.orderByStrings = new Array(vm.orderOptions.length);
   var i=0;
   if(vm.orderOptions.length>0) {
      vm.orderByStrings[0] = " "+vm.orderOptions[0]+" (DESC)";
   for(i=1; i<vm.orderOptions.length; i++)
      vm.orderByStrings[i] = " "+vm.orderOptions[i]+" (ASC)";
   }


   vm.orderItemsBy = function(attribute, index) {
      if(vm.orderBy==attribute) {
         vm.orderBy='-'+attribute; //If items are already ordered by attribute, order them in descending order
         if(vm.orderBy=='-'+attribute) vm.orderByStrings[index] = ' '+attribute+" (ASC)";
         else vm.orderByStrings[index] = ' '+attribute+" (DESC)";
      }
```

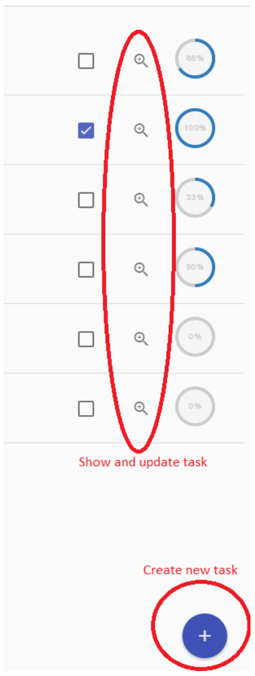
```
        else {  var str = vm.orderBy.replace('-', '');
              var ind = vm.orderOptions.indexOf(str);
                 if (ind != -1) {
                 vm.orderByStrings[ind] = " "+str+" (ASC)";
                 }
              vm.orderBy = attribute;
              vm.orderByStrings[index] = ' '+attribute+" (DESC)";
        }


     }


  }
```

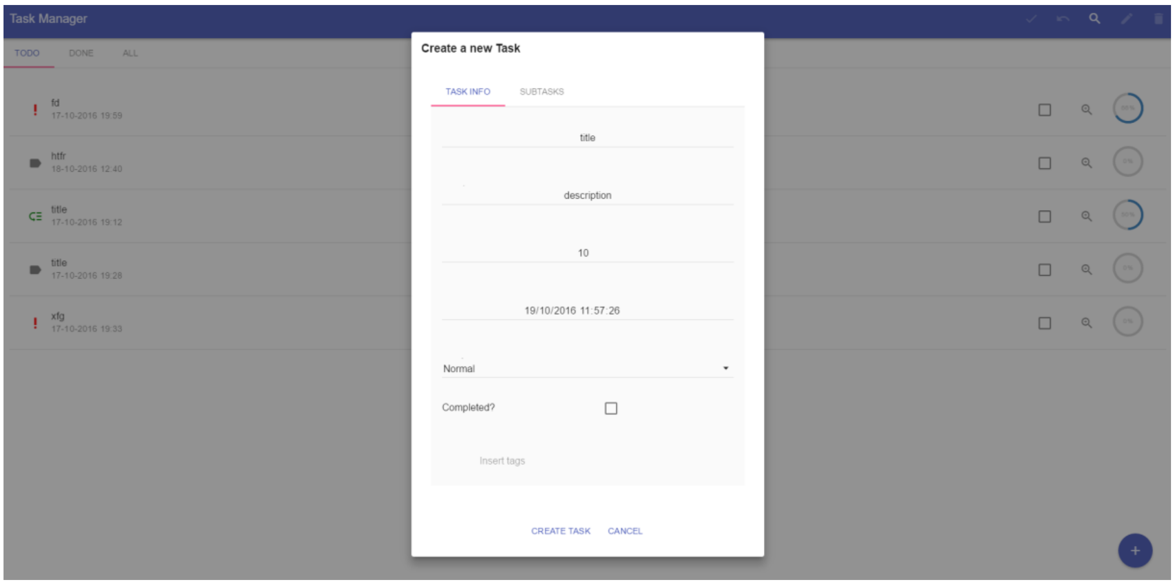
The sorting options are passed to the directive through the attribute ‘orderOptions’, so the sorting feature can potentially be used without any change to the customToolbar directive’s code also to sort items by other attributes.

```
'<md-menu-item ng-repeat="option in customToolbarCtrl.orderOptions">'+
               '<md-button ng-click="customToolbarCtrl.orderItemsBy(option, $index)">Order by
{{customToolbarCtrl.orderByStrings[$index]}}</md-button>'+
               '</md-menu-item>'+
…
```

3.4 Creation and Update interfaces

The creation interface of the app, originally including only the task’s title, has been extended to include all the attributes of the tasks. Both the creation and the update interfaces have been realized by using the $md-dialog service with a custom template. The button to create a new task is the same of the original project, while every task has a specific button (‘show task’) to open the update interface for the task. The interface will show the current values of the task’s attributes and will let the user change them.





fd
17-10-2016 19:59

htfr
18-10-2016 12:40

title
17-10-2016 19:12

title
17-10-2016 19:28

xfg
17-10-2016 19:33

Create a new Task

title

description

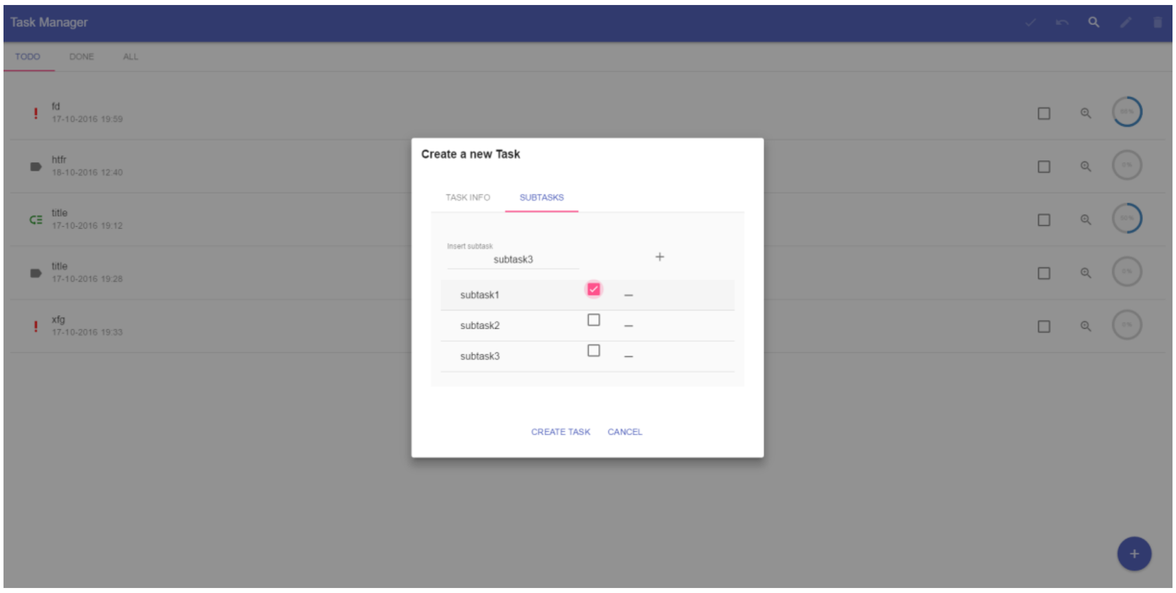
10

19/10/2016 11:57:26

Normal

Completed? ☐

Insert tags



fd
17-10-2016 19:59

htfr
18-10-2016 12:40

title
17-10-2016 19:12

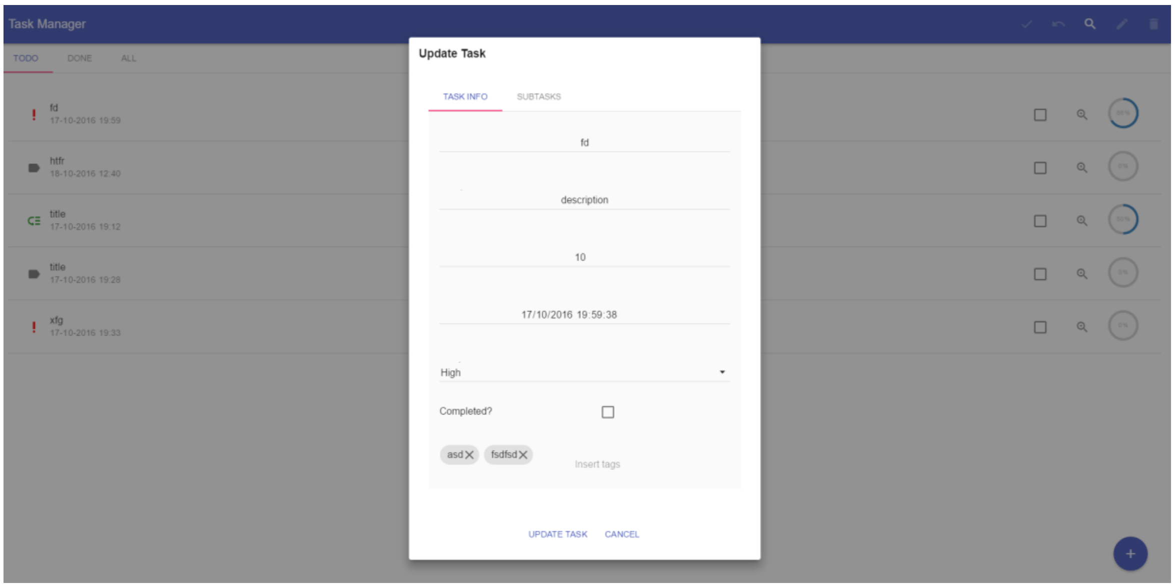
title
17-10-2016 19:28

xfg
17-10-2016 19:33

Create a new Task

Insert subtask
subtask3 +

subtask1	✓	–
subtask2	☐	–
subtask3	☐	–



fd
17-10-2016 19:59

htfr
18-10-2016 12:40

title
17-10-2016 19:12

title
17-10-2016 19:28

xfg
17-10-2016 19:33

Update Task

fd

description

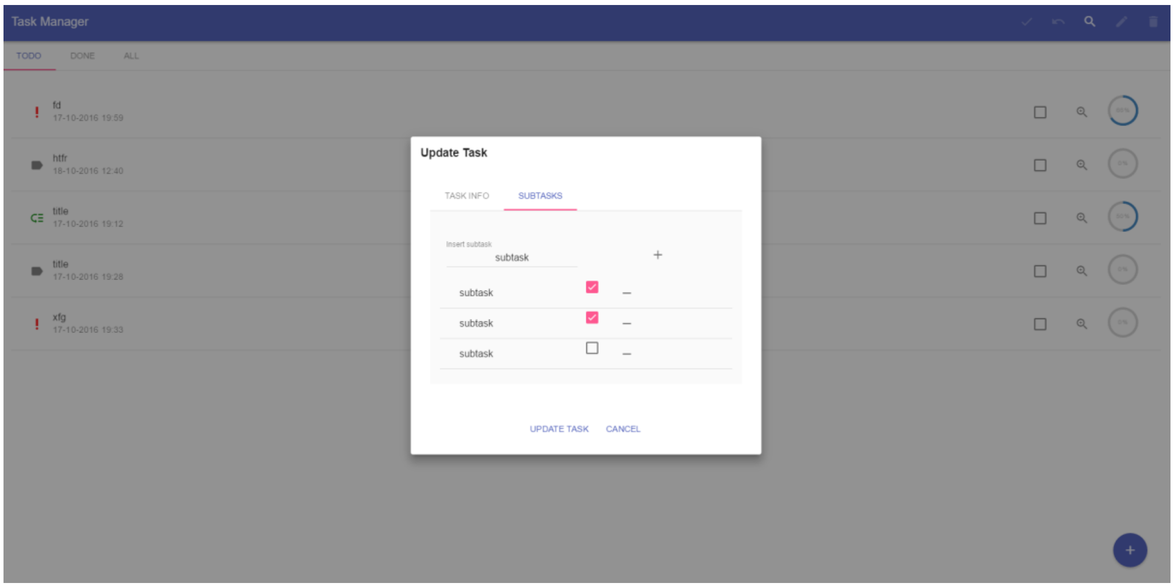
10

17/10/2016 19:59:38

High

Completed? ☐

asd ✕ fsdfsd ✕ Insert tags



The content part of the md-dialog is the same for the creation and update interfaces, and it is showed through the directive formDialog, which takes as attributes the task, the form name and the title of the dialog.

```
angular
    .module('todoApp')
    .directive('formDialog', directive);

  function directive() {
    return {
      scope: {},
      bindToController: {
        task: '=',
        formName: '=',
        title: '@',
      },
...
```

The directive’s template contains a directive md-tabs with two tabs, one to create/remove/change the status of the task’s subtasks, the other to set the other attributes.
The subtasks have a name and a status, which is connected to the task’s status, as in, if the task is set to done, then the status of all subtasks is set to done, vice versa if a task, previously set to

done, is set to not done, then the subtasks' status is set to not done (this is true even when the task's status is changed outside the creation and update interfaces).
Similarly if the status of all the subtasks is set to done or if a new subtask is created or if the status of an existing one is set to not done, the task's status is changed accordingly.

```
//Function to add subtask
      vm.addSubtask = function() {
         vm.task.subtasks.push({
            title : vm.subtask,
            done : false
         });
         vm.task.done=false;
      }
//Function called to check if all subtasks are done. If yes, task is set to done.
      vm.checkStatusChange = function() {
         var length = vm.task.subtasks.length;
         if(length>0) {
         var done = true;
         var i=0;
         for(i=0; i<length; i++) {
               done = done && vm.task.subtasks[i].done;
               if(!done) break;
         }
         vm.task.done = done;
      }
      }
//Function called to delete subtask.
      vm.deleteSubtask = function(item) {
         var index = vm.task.subtasks.indexOf(item);
                        if (index != -1)
                           vm.task.subtasks.splice(index, 1);
         vm.checkStatusChange();

      }

//Function called when task status is changed. If task set to done => all subtasks set to done. If set to not
done => all subtasks set to not done.
      vm.setSubtasksStatus = function() {
               var i=0;
        if(vm.task.done) {

            for(i=0; i<vm.task.subtasks.length; i++)
                vm.task.subtasks[i].done=true;
        } else {
            for(i=0; i<vm.task.subtasks.length; i++)
              vm.task.subtasks[i].done=false;
        }

      }
```

The creation, deletion and update of the task’s tags are handled through the directive md-chips.
A max length has been set for the title, description and subtasks’ name input fields.
The actions part of the md-dialog are different for the two interfaces and they are handled inside the two functions of the ToDo controller called when the designed buttons are clicked.

```
        '<md-dialog flex="30" aria-label="Add Task">' +
        '<form name="addTaskForm" novalidate>'+
   '<form-dialog task="task" form-name="addTaskForm" title="Create a new Task"></form-dialog>'+
        '  <md-dialog-actions class="md-padding" layout="row" layout-align="center center">' +
         '    <md-button type="submit" ng-disabled="addTaskForm.$invalid" class="md-primary" ng-
click="insert()" >' +
        '     Create task' +
        '    </md-button>' +
        '    <md-button class="md-primary" ng-click="closeDialog()" >' +
        '      Cancel' +
        '    </md-button>' +
        '  </md-dialog-actions>' +
       '</form>'+
        '</md-dialog>',
       locals: {
        task: vm.task
       },...

  '<md-dialog flex="30" aria-label="Update Task" >' +
        '<form name="updateTaskForm" novalidate>'+
        '<form-dialog task="task" form-name="updateTaskForm" title="Update Task"></form-dialog>'+

      '  <md-dialog-actions class="md-padding" layout="row" layout-align="center center">' +

         '    <md-button type="submit" ng-disabled="updateTaskForm.$invalid" class="md-primary" ng-
click="update(item)" class="md-primary">' +
        '    Update task' +
        '    </md-button>' +
        '    <md-button class="md-primary" ng-click="closeDialog()" class="md-primary">' +
        '      Cancel' +
        '    </md-button>' +
        '  </md-dialog-actions>' +
         '</form>'+
          '</md-dialog>',
        locals: {
          task: vm.task
        },…

  <md-button class="md-fab md-primary add-button" aria-label="Add Todo" ng-
click="ctrl.addTask($event)">
                <md-icon>add</md-icon>
             </md-button>…

<md-button ng-click="customListCtrl.showFunction($event, item)" class="md-icon-button" aria-
label="Show Task"><md-tooltip>Show task</md-tooltip><md-icon>zoom_in</md-icon></md-button>…
```

3.5 Completion Percentage

The completion percentage of the task is calculated every time a task is created or updated. For a task without subtasks, the completion percentage is 0% if the task is set to not done, 100% if it is set to done. For a task with subtasks it is equal to Math.floor((done_subtasks/number_of_subtasks)*100).

```
 //Calculates task's completion percentage
    vm.calculateCompletion = function(subtasks) {
      var length = subtasks.length;
      if(length>0) {
      var count=0;
      var i=0;
      for(i=0; i<length; i++)
         if(subtasks[i].done) count++;

      return Math.floor((count/length)*100);
      } else return 0;
    }

    //Creates a new item with the given parameters
    vm.createItem = function(title,description, priority, done, date, estimatedWork, tags, subtasks) {
     var completion;
      if(done) completion = 100;
      else
         if(subtasks!=null)
            completion = vm.calculateCompletion(subtasks);
         else completion = 0;
    …
```

The completion percentage of the tasks is showed by using the percentageCircle's directive, that through the use of the classes defined in the file circle.css, shows a percentage circle for each task.