CS5003 - Project 1: Bivouac - Report

Student number: 130018883 Words: 687

Introduction

This project was to develop a project management application using the Google Drive service to share and manage updates. This included using the Google Drive Realtime API so that updates are seen in real time by all users simultaneously.

Model-view-controller

My implementation of this project followed the model-view-controller pattern, utilising the inherent model associated with a Google Drive Realtime File. The model I chose is illustrated in Figure 1 in the Appendix, and is as follows:

In the Collaborative Map associated with the document I added two key-value pairs; one Collaborative String for the project title, and one Collaborative List for the tasks included in this project. Each entry in the list is a Collaborative Map that represents a task, and currently has four key-value pairs; two Collaborative Strings, one for the name of the task and one for the due date, and two primitive strings for the priority and status of the task. If the project were to be expended, more key-value pairs could be added.

Functionality

When **initializeModel()** is called, it initialised the model by creating the project title, task list and a single task, and associating them as per the described model.

The **onFileLoaded()** function takes the information from the model in the file and presents it on the page, by binding the project title string with the text area on the page and calling the **addTask()** function for each task in the model. It also adds an event listener to call **addTask()** if any new tasks are added, adds event handlers to the buttons, and loads the users other files into a drop-down menu for selection.

In order to add each task to the page, **addTask()** builds a table using DOM elements. The strings for task name and due date are bound to text fields so that updates are kept in sync, and priority and status are displayed as drop-down menus. Event listeners are added so that the model is updated when the option is changed on the page, and the page is changed if the data in the model is altered.

To create a new task, the **createTask()** function is called. This is similar to code in **initializeModel()**; adding a new Collaborative Map to the task list with the four properties as per the model.

The **loadOtherProjects()** function takes the files returned by **rtclient.listFiles()** and adds them as options in a drop-down menu. The current file ID is read from the URL and excluded. These can then be opened using the **loadProject()** function, which redirects to the URL with the specified file ID to open and view the project

Finally, the **shareProject()** function utilises the the Drive API to insert a permission into the files ACL. The email address of the desired collaborator is taken from a field in the page, and validated using some JavaScript taken from W3Schools at http://www.w3schools.com/js/js_form_validation.asp.

To do

There are some areas which I would like to expand on, given more time or familiarity with the Google Drive APIs. These are marked with 'TODO' comments in the code, and are as follows:

Currently, alternate projects are identified by their file ID, and chosen and redirected to in this way. This is due to the fact that the project title is stored as a key-value pair in the document model, and is not associated with the actual filename. I explored the Drive API in the hopes of finding a way to update the filename with the model, although couldn't find a way to implement this. The next step would be to add this functionality, identify the users other projects by name and allow navigation of these projects by title.

Inserting a permission is functionally correct, but is not a good experience on the recipients side. A generic Google Drive sharing email is received, from which the user must take the file ID and visit the correct URL manually, or open the drop-down menu of an existing project, to visit the file. Creating a better collaboration system would be another next step, which could include a custom email to the collaborator with a valid link for quicker access.

Appendix:

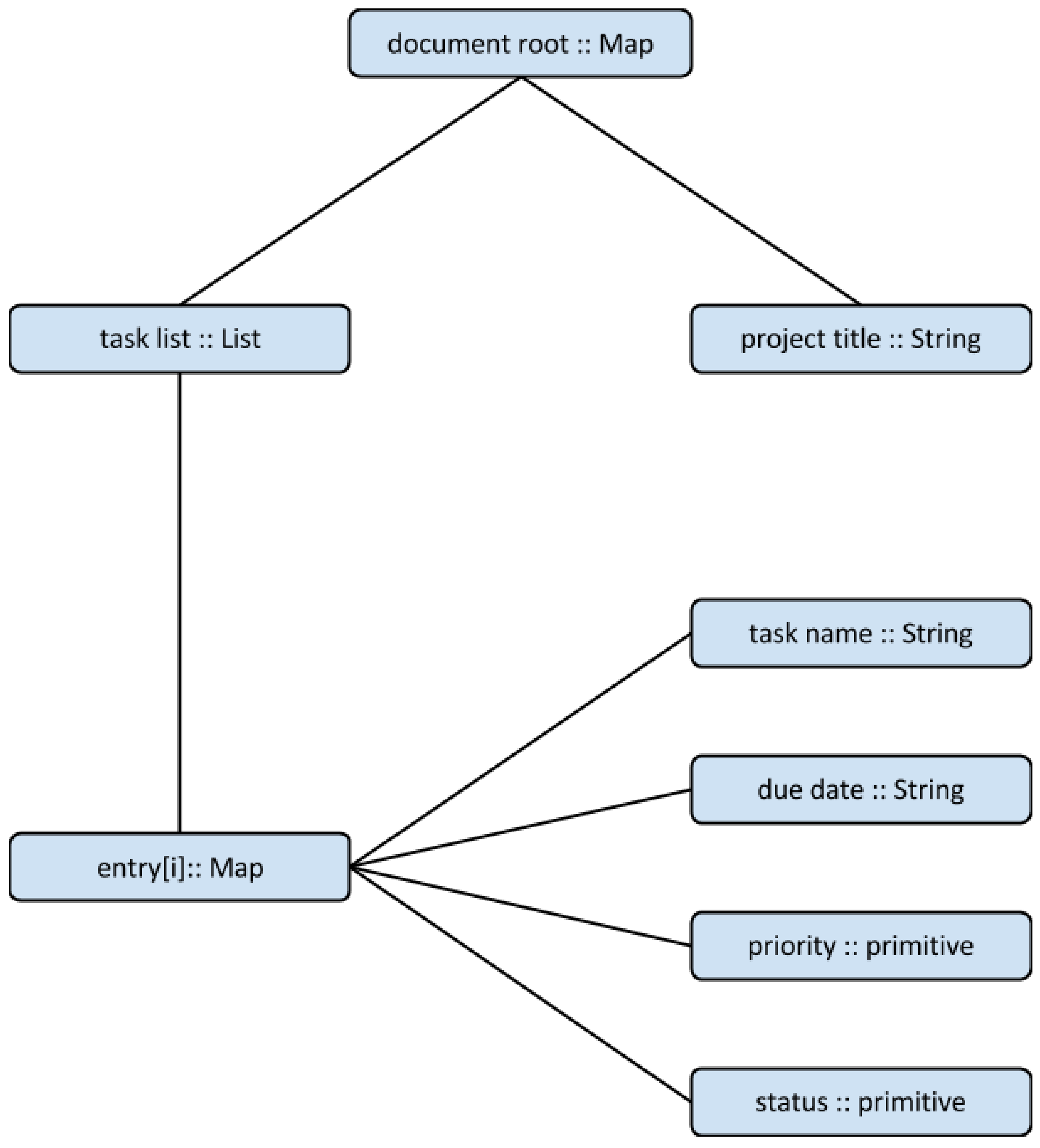

Figure 1