Advanced Programming 2024
Task Oriented Programming, Part 2/2
Assignment 13

May 28, 2024

1 Goal

The goal of this exercise is to familiarize yourself with the use of Shared Data Sources and distributed systems with the iTask system. This is done by creating a distributed application in which group members can create tasks for each other and select and execute one of their tasks after logging in.

On Brightspace you find a Clean main module, `TOP2.icl`, that contains a suggestion for the top-level structure of your application, and a Clean definition module, `Job.dcl`, that contains the `Job` type definition relevant to this assignment. Create the corresponding `Job.icl` module.

Please note that this assignment is underspecified and offers you a lot of design freedom. Feel free to alter the provided definitions and structure. We advise you to develop all domain data types and pure functions in `Job` and all tasks in `TOP2`.

2 Assignment

Module `TOP2` contains the boilerplate code to get started with a multi-user, multi-task system.

```
Start :: *World -> *World
Start world
 = doTasks
     { WorkflowCollection
     | name           = // a name for your application
     , workflows      = [ workflow "Manage users" "manage users"              manageUsers
                        , workflow "View members" "view current group member" viewMembers
                        // the list of tasks
                        ]
     , loginMessage   = // optional login message
     , welcomeMessage = // optional welcome message
     , allowGuests    = False // this application does not make sense for anonymous users
     } world

viewMembers :: Task [User]
viewMembers
 = viewSharedInformation [] users
```

Your solution should use at least the following persistent SDSs:

```
openJobsSDS :: SimpleSDSLens [Job]
openJobsSDS = sharedStore "openjobs" []

doneJobsSDS :: SimpleSDSLens [(UserId,Job)]
doneJobsSDS = sharedStore "donejobs" []
```

```
neglectedJobsSDS :: SimpleSDSLens [Job]
neglectedJobsSDS = sharedStore "neglectedjobs" []
```

You can test your application by creating several browser windows (or tabs) and direct each of them to `localhost:8080`. In each of these windows (or tabs) log in as another user. The generated web server application handles all events appropriately and generates the specified interface for each user.

Extend the `workflows` struct member of the `WorkflowCollection` argument of the `Start` function with the following tasks that group members can perform. Please use the suggested names for the task functions.

1. the task `viewAllJobs` displays all open jobs of all group members (the content of `openJobsSDS`)
2. the task `viewDoneJobs` displays all jobs that have been performed (the content of `doneJobsSDS`)
3. the task `viewNeglectedJobs` displays all jobs which deadline has passed and have been stored as such (the content of `neglectedJobsSDS`)
4. the task `viewMyJobs` displays all tasks of the current user (the iTask SDS `currentUser` contains the current user)
5. the task `createJob` first lets the user create a *non-empty* selection of group members, followed by a *non-empty* description of the job to do, followed by an *optional* deadline, and adds it to `openJobsSDS`.
6. the task `doAJob` lets the user select one of her jobs that either have no deadline or a deadline in the future and let her confirm that that job has been done. After the job has been done, it should be removed from `openJobsSDS` and added to `doneJobsSDS` in *one atomic update.*
7. the task `cleanJobs` removes all jobs with a passed `?Just deadline` from `openJobsSDS` and adds them to `neglectedJobsSDS` in *one atomic update.*