Advanced Programming (I00032) 2016
iTasks combinators

Assignment 6

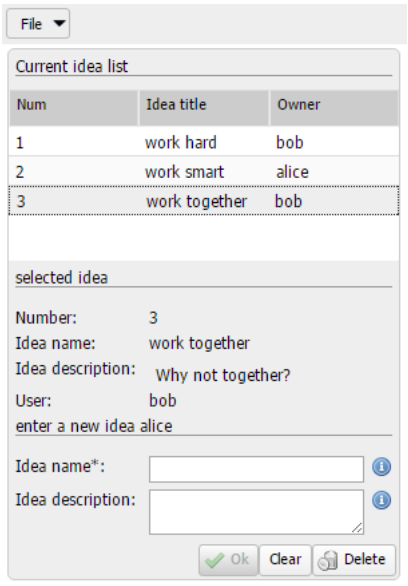
Goal

After making this exercise you should be able to make multi-sure iTask applications. You can use Shared Data Sources to make a persistent state that is shared by the users. You can use combinators to compose tasks to more complex tasks.

1 A Multi-user Idea Box

In this exercise you make a multi-user idea box. Like always it is good to construct an iTask program incrementally. Start with a very simple and limited version of the program, for instance only an editor for an ideas, and add features one-by-one. Of course it is wise to think ahead, for instance by introducing the share to store ideas early in the development.

The interface of the final program for user can look like:



Each user can make ideas of type

```
:: Idea =
    { idea_name :: String
    , idea_description :: Maybe Note
    }
```

Your iTask program adds the name of the `user` and a `number` to such an idea.

```
:: NamedIdea =
  { number :: Int
  , idea   :: Idea
  , user   :: String
  }
```

To access these ideas from different users the list of existing ideas is stored in a SDS.

```
ideas :: Shared [NamedIdea]
ideas = sharedStore "Ideas" []
```

When a user starts your program in a browser window it first asks the name of the user. Different users can login simultaneously from different browser windows, or tabs.

After providing her name the user gets a an interface as shown in the picture above. There is a table with existing ideas that connot be changed. The table with existing ideas is obtained by using `enterChoiceWithShared` instead of the `viewSharedInformation` you might have expected.

Note that the table does not display the description of the idea; just the number, name, and user are shown. This is obtained by the option `ChooseWith (ChooseFromGrid conv)` of `enterChoiceWithShared` where `conv` is an appropriate conversion function.

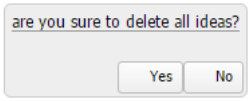
When you like you can show the full details of the selected idea with using the operator `>&^`. This operator feeds the result of the task on its lefthand-side (the `enterChoiceWithShared`) as a read-only share to the task on the righthand-side (a `viewSharedInformation`). The task on the lefthand-side stays active. This implies that we can select different ideas over and over again.

Using the step operator `>>*`, the program offers four options to the user:

1. The `Ok` button can be pressed when the idea name is at least 3 characters long and does not occur as name in the list of existing ideas. You will probably need the `upd` function for shares.

2. The `Clear` button removes all information in the current editor. The user can of course direct start with adding a new idea.

3. The `delete` button removes all existing ideas from the entire list. Add a confirm-box with buttons `Yes` and `No` to check is the user is sure to delete all ideas.



4. Finally, there is the `ActionQuit`. By design this is shown as an item in the pulldown menu `File` in the left upper corner of the window.

When the user enters the name `admin` when the program starts she gets an ordinary editor for the lists of ideas. In this way the administrator can fix all problems with the list of ideas that might occur dynamically.

Deadline

The deadline for this exercise is October 11, 2016, 13:30h (just before the next lecture).