Thème 1 - TD3 - Modélisation avec les Types (3/4)

Sorbonne Université - Master Informatique M1 - STL

MU4IN510 Programmation Avancée en Fonctionnel - 2021

Dans ce TD nous allons reconstruire, de façon partielle et simplifiée, le modèle d'un logiciel de suivi de projet du type *Trello*.

Pour cela, nous allons utiliser le langage des types de Haskell, avec en particulier les ensembles de `Data.Set` [1], les tables associatives de `Data.Map` [2] et les séquences de `Data.Sequence` [3]. Pour tous les types définis on explicitera un (ou plusieurs) invariant(s) que les opérations sur ces types devront respecter.

L'objectif est de traduire le cahier des charges qui suit par des définitions de type, y compris pour les fonctionnalités (qui ne seront pas implémentées et resteront donc `undefined`). Pour ces dernières on indiquera des préconditions et postconditions éventuelles.

Remarque : en modélisation, on pourra utiliser des notations mathématiques/logiques pour les propriétés (invariants, pré/post-conditions), leur traduction en Haskell étant facultative à ce state.

Nous rappelons les principes de modélisation discutés en cours :

- Principe 1 : les entités (du domaine) sont des sommes de produits (éventuellement records)
- Principe 2 : les états remarquables ont des constructeurs dédiés
- Principe 3 : les états incohérents ne sont pas constructibles (*smart constructors*)
- Principe 4 : les invariants des entités sont définis explicitements
- Principe 5 : les événements (opérations) sont des fonctions, avec préconditions et postconditions éventuelles

Cahier des charges : projet Projective

Projective est un logiciel de suivi de projet de développement logiciel.

1) Architecture

Système

Dans le cadre d'un **système**, des **projets** sont développés par des **membres** de projet. Chaque membre du système, œuvrant (ou non) sur certains projets, est identifié par un identifiant unique numérique. Chaque projet est également identifié de façon similaire.

Le système initial n'a pas de membre ni de projet.

Membres

Les informations concernant un membre de projet (au sein d'un système) sont les suivantes :

- prénom
- nom
- login
- email

1. http://hackage.haskell.org/package/containers-0.6.2.1/docs/Data-Set.html
2. http://hackage.haskell.org/package/containers-0.6.2.1/docs/Data-Map.html
3. http://hackage.haskell.org/package/containers-0.6.2.1/docs/Data-Sequence.html

Projets

Un informations concernant un projet sont les suivantes :

— le descriptif textuel du projet (une/deux lignes de texte)
— les **membres** actifs
— les **tableaux** actifs

Chaque tableau est identifié par un identifiant unique.

Tableaux

Un tableau est simplement un séquencement de **cartes**.

Cartes

Les informations d'une carte sont les suivantes :

— le nom (court) de la carte
— un descriptif textuel
— les membres actifs pour cette carte
— la séquence de **tâches** à réaliser

Tâches

Les informations d'une tâche sont les suivantes :

— descriptif textuel de la tâche
— date d'activation de la tâche (`timestamp`)
— membres affectés à cette tâche
— état de la tâche parmi : non-affectée, en cours (running) avec date limite (deadline, éventuellement "illimitée"), ou complétée à telle date

Remarque : on ne s'intéressera pas aux manipulations des dates, qui resteront donc abstraites.

2) Fonctionnalités

Concernant les fonctionnalités, on utilisera des types fonctionnels pour spécifier les opérations, avec des préconditions éventuelles. On pourra implémenter directement l'opération si cela est faisable (en supposant les préconditions), sinon on laissera l'opération `undefined`. Dans tous les cas, on pourra proposer des postconditions pour caractériser au mieux l'opération effectuée.

De plus, les opérations seront modélisées de façon ascendante (*bottom-up*), c'est-à-dire en commençant par les types de plus bas niveaux dans le système (exemple tâche est considéré avant carte).

Voici les principales fonctionnalités attendue de l'application :

— ajout d'un membre / retrait d'un membre
— ajout d'un projet / retrait d'un projet
— création / suppression d'un tableau
— ajout / suppression d'une carte à un tableau
— réordonnancement des cartes d'un tableau
— modifier les affectations de membres à une carte
— changer le status d'une carte (complétion)

D'autres opérations seront éventuellement définies en fonction des besoins.