```haskell
{-# LANGUAGE OverloadedStrings #-}
{-|
  This is the Haskell implementation of the Datatypes
  described in the UML-class-diagram from storedData.pdf.
  It is also converted into a .pdf and included into storedData.pdf.
|-}
module OpenBrain.Data where

import Data.Aeson ((.=), ToJSON(..), object)
import Data.Function (on)
import Happstack.Server (FromReqURI(..))
import qualified Data.Aeson as Aeson

import OpenBrain.Data.Id
import OpenBrain.Data.Hash
import OpenBrain.Data.Json
import OpenBrain.Data.Salt

data Description = Description {
  descriptionId :: DescriptionId
, author        :: Author
, headline      :: Headline
, description   :: String
, creationTime  :: Timestamp
, deletionTime  :: Timestamp
} deriving (Show)

data Article = Article {
  articleId    :: ArticleId
, content      :: String
, children     :: [ArticleId]
, aDescription :: Description
} deriving (Show)

data Relation = Relation {
  relationId   :: RelationId
, source       :: ArticleId
, target       :: ArticleId
, rType        :: RelationType
, rDescription :: Description
} deriving (Show)

data RelationType = RelationAttack | RelationDefense
  deriving (Show, Read, Eq, Ord, Enum)

data Collection = Collection {
  collectionId :: CollectionId
, articles     :: [ArticleId]
, cDescription :: Description
} deriving (Show)

data Discussion = Discussion {
  discussionId :: DiscussionId
, participants :: [UserId]
, deadline     :: Timestamp
, weights      :: [(UserId, Weight, RelationId)]
, result       :: Maybe Result
, dCollection  :: Collection
} deriving (Show)

data Result = Result {
  resultId :: ResultId
, choices  :: [(CollectionId, Votes)]
, voters   :: [(UserId, Voted)]
} deriving (Show)

data User = User {
  userId       :: UserId
, username     :: String
, userhash     :: Hash
, usersalt     :: Salt
, userCreation :: Timestamp
, lastLogin    :: Timestamp
, isAdmin      :: Bool
, profile      :: Maybe ArticleId
, session      :: Maybe SessionKey
} deriving (Show)

{-| Type aliases: |-}
type Author       = UserId
type Count        = Int
type Headline     = String
type Heir         = UserId
type IsAdmin      = Bool
type Limit        = Int
type Offset       = Int
type SessionKey   = String
type Timestamp    = String
type Username     = String
type Voted        = Bool
type Votes        = Int
type Weight       = Int

{-| Instances of Eq: |-}
instance Eq Description where
  (==) = (==) `on` descriptionId
instance Eq Article where
  (==) = (==) `on` articleId
instance Eq Relation where
  (==) = (==) `on` relationId
instance Eq Collection where
  (==) = (==) `on` collectionId
instance Eq Discussion where
  (==) = (==) `on` discussionId
instance Eq Result where
  (==) = (==) `on` resultId
instance Eq User where
  (==) = (==) `on` userId

{-| Instances of Ord: |-}
instance Ord Description where
  compare = compare `on` descriptionId
instance Ord Article where
  compare = compare `on` articleId
instance Ord Relation where
  compare = compare `on` relationId
instance Ord Collection where
  compare = compare `on` collectionId
instance Ord Discussion where
  compare = compare `on` discussionId
instance Ord Result where
  compare = compare `on` resultId
instance Ord User where
  compare = compare `on` userId

{-| Instances of ToJSON: |-}
instance ToJSON Description where
  toJSON d = object [
      "descriptionId" .= descriptionId d
    , "author"        .= author        d
    , "headline"      .= headline      d
    , "description"   .= description   d
    , "creationTime"  .= creationTime  d
    , "deletionTime"  .= deletionTime  d
    ]
instance ToJSON Article where
  toJSON a = merge (toJSON $ aDescription a) o
    where
      o = object [
          "articleId" .= articleId a
```

```haskell
        , "content"   .= content   a
        , "children"  .= children  a
        ]
instance ToJSON Relation where
  toJSON r = merge (toJSON $ rDescription r) o
    where
      o = object [
          "relationId" .= relationId r
        , "source"     .= source     r
        , "target"     .= target     r
        , "rType"      .= rType      r
        ]
instance ToJSON RelationType where
  toJSON = toJSON . show
instance ToJSON Collection where
  toJSON c = merge (toJSON $ cDescription c) o
    where
      o = object ["collectionId" .= collectionId c, "articles" .= articles c]
instance ToJSON Discussion where
  toJSON d = merge (toJSON $ dCollection d) o
    where
      o = object [
          "discussionId" .= discussionId d
        , "participants" .= participants d
        , "deadline"     .= deadline     d
        , "weights"      .= weights      d
        , "result"       .= result       d
        ]
instance ToJSON Result where
  toJSON r = object [
      "resultId" .= resultId r
    , "choices"  .= choices  r
    , "voters"   .= voters   r
    ]
instance ToJSON User where
  toJSON u = object [
      "userId"       .= userId       u
    , "username"     .= username     u
    , "userCreation" .= userCreation u
    , "lastLogin"    .= lastLogin    u
    , "isAdmin"      .= isAdmin      u
    , "profile"      .= profile      u
    ]
{-| Instances of FromReqURI |-}
instance FromReqURI RelationType where
  fromReqURI "RelationAttack"  = Just RelationAttack
  fromReqURI "RelationDefense" = Just RelationDefense
  fromReqURI _ = Nothing
```