```haskell
{-# LANGUAGE ExistentialQuantification, GADTs #-}
module OpenBrain.Backend.DSL where

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

{-| The BackendDSL and it's verbs: |-}
data BackendDSL r where
  -- | Composition:
  Backendλ :: BackendDSL p -> (p -> BackendDSL r) -> BackendDSL r
  Nop      :: r -> BackendDSL r
  -- | User related:
  AddUser    :: Username -> (Hash, Salt) -> IsAdmin -> BackendDSL (Maybe UserId)
  DeleteUser :: UserId -> Heir -> BackendDSL ()
  GetNobody  :: BackendDSL UserId
  GetUser    :: UserId -> BackendDSL User
  HasUser    :: Username -> BackendDSL (Maybe UserId)
  Login      :: UserId -> (Salt -> Hash) -> BackendDSL (Maybe SessionKey)
  Validate   :: UserId -> SessionKey -> BackendDSL Bool
  Logout     :: UserId -> BackendDSL ()
  SetAdmin   :: UserId -> IsAdmin -> BackendDSL ()
  SetPasswd  :: UserId -> (Salt -> Hash) -> BackendDSL ()
  SetProfile :: UserId -> Maybe ArticleId -> BackendDSL ()
  -- | Description related:
  AddDescription    :: Author -> Headline -> String -> BackendDSL NewDescriptionId
  DeleteDescription :: DescriptionId -> BackendDSL ()
  GetDescription    :: DescriptionId -> BackendDSL Description
  SetHeadline       :: DescriptionId -> Headline -> BackendDSL ()
  SetDescription    :: DescriptionId -> String -> BackendDSL ()
  -- | Article related:
  AddArticle :: NewDescriptionId -> String -> BackendDSL ArticleId
  Clone      :: ArticleId -> BackendDSL ArticleId
  GetArticle :: ArticleId -> BackendDSL Article
  SetContent :: ArticleId -> String -> BackendDSL ()
  -- | Relation related:
  AddRelation :: NewDescriptionId -> RelationType -> ArticleId -> ArticleId -> BackendDSL RelationId
  GetRelation :: RelationId -> BackendDSL Relation
  -- | Collection related:
  AddCollection   :: NewDescriptionId -> [ArticleId] -> BackendDSL NewCollectionId
  CollectArticles :: CollectionId -> [ArticleId] -> BackendDSL ()
  ForgetArticles  :: CollectionId -> [ArticleId] -> BackendDSL ()
  GetCollection   :: CollectionId -> BackendDSL Collection
  -- | Discussion related:
  AddDiscussion  :: NewCollectionId -> [UserId] -> Timestamp -> BackendDSL DiscussionId
  GetDiscussion  :: DiscussionId -> BackendDSL Discussion
  SetParticipant :: DiscussionId -> UserId -> Bool -> BackendDSL ()
  Weight         :: DiscussionId -> UserId -> Weight -> RelationId -> BackendDSL ()
  -- | Result related:
  AddResult :: DiscussionId -> [CollectionId] -> BackendDSL ResultId
  GetResult :: ResultId -> BackendDSL Result
  Vote      :: ResultId -> UserId -> CollectionId -> BackendDSL ()
  -- | Paging:
  ArticleCount     :: BackendDSL Count
  CollectionCount  :: BackendDSL Count
  DescriptionCount :: BackendDSL Count
  DiscussionCount  :: BackendDSL Count
  RelationCount    :: BackendDSL Count
  ResultCount      :: BackendDSL Count
  UserCount        :: BackendDSL Count
  PageArticles     :: Limit -> Offset -> BackendDSL [ArticleId]
  PageCollections  :: Limit -> Offset -> BackendDSL [CollectionId]
  PageDescriptions :: Limit -> Offset -> BackendDSL [DescriptionId]
  PageDiscussions  :: Limit -> Offset -> BackendDSL [DiscussionId]
  PageRelations    :: Limit -> Offset -> BackendDSL [RelationId]
  PageResults      :: Limit -> Offset -> BackendDSL [ResultId]
  PageUsers        :: Limit -> Offset -> BackendDSL [UserId]

```

```haskell
{-| The Monad instance for BackendDSL to enable beautiful composition. |-}
instance Monad BackendDSL where
  (>>=)  = Backendλ
  return = Nop

{-|
  A BackendProcessor, which must be supplied by OpenBrain.Backend.Load from the Config file.
  This procedure makes sure, that the rest of the Application only uses the BackendDSL to communicate
  with the Backend and the BackendProcessor stays exchangable as long as the interpretation
  of the DSL doesn't change between Processors.
|-}
class BackendProcessor b where
  process :: b -> BackendDSL r -> IO r

{-| A Container for BackendProcessors: |-}
data CBackendProcessor = forall b . BackendProcessor b => CBackendProcessor b
instance BackendProcessor CBackendProcessor where
  process (CBackendProcessor b) = process b
```