**Coordinación de Ingeniería en Sistemas Computacionales**

| | |
|---|---|
| **Materia:** | 1S7B-PROGR.LOG.FUNC |
| **Nombre Informe:** | Investigación: Las librerías para el multipropósito en Haskell |
| **Alumno(s):** | Pizano Zamora Leonardo Gabriel |

## Objetivo

El objetivo de esta investigación es explorar el uso de la biblioteca Persistent en Haskell para interactuar con bases de datos relacionales, comprendiendo su funcionamiento, ventajas y cómo se integra con el lenguaje Haskell para simplificar el acceso y la manipulación de datos.

## Marco Teórico

Persistent es una biblioteca de mapeo objeto-relacional (ORM) en Haskell que permite a los desarrolladores interactuar con bases de datos relacionales de una manera tipo Haskell. Proporciona una capa de abstracción sobre la base de datos, lo que facilita la definición de modelos de datos y la realización de consultas.

## Desarrollo

Definición de Modelos: Con Persistent, los modelos de datos se definen utilizando un lenguaje tipo DSL (Domain Specific Language) dentro de Haskell. Por ejemplo, para definir un modelo de datos para una tabla de usuarios en una base de datos PostgreSQL, se puede escribir:

```
{-# LANGUAGE TemplateHaskell #-}
{-# LANGUAGE QuasiQuotes #-}
{-# LANGUAGE TypeFamilies #-}
{-# LANGUAGE GADTs #-}
{-# LANGUAGE GeneralizedNewtypeDeriving #-}
import Database.Persist.TH

share [mkPersist sqlSettings, mkMigrate "migrateAll"] [persistLowerCase|
User
        name String
        age Int
        deriving Show
|]
```

Interacción con la Base de Datos: Una vez que se han definido los modelos de datos, Persistent permite interactuar con la base de datos de manera tipo Haskell. Por ejemplo, para insertar un nuevo usuario en la base de datos, se puede escribir:

```
import Database.Persist.Postgresql
```

```
import Control.Monad.IO.Class (liftIO)

main :: IO ()
main = runSqlite "test.db" $ do
        runMigration migrateAll
        insert $ User "Alice" 30
        insert $ User "Bob" 25
```

Consultas: Persistent facilita la realización de consultas a la base de datos utilizando una interfaz tipo DSL en Haskell. Por ejemplo, para recuperar todos los usuarios mayores de 25 años, se puede escribir:

```
import Database.Persist
import Database.Persist.Sqlite
import Control.Monad.IO.Class (liftIO)

main :: IO ()
main = runSqlite "test.db" $ do
        users <- selectList [UserAge >. 25] []
        liftIO $ print users
```

## Evaluación

Persistent simplifica el acceso y la manipulación de bases de datos en Haskell al proporcionar una interfaz tipo Haskell para definir modelos de datos, realizar consultas y realizar operaciones de manipulación de datos. Su integración con Haskell hace que sea fácil de usar y reduce la cantidad de código boilerplate necesario para interactuar con la base de datos.

## Conclusiones}

En conclusión, Persistent es una herramienta poderosa para trabajar con bases de datos en Haskell. Facilita la definición de modelos de datos, la realización de consultas y la manipulación de datos, lo que permite a los desarrolladores centrarse en la lógica de la aplicación en lugar de en los detalles de implementación de la base de datos.