Práctica #3: Introducción a Haskell y Proyecto App Haskell

Montes Solis Kimberly

── .✦ Introducción

En esta práctica se explorará el lenguaje de programación **Haskell**, su funcionamiento, características principales, herramientas básicas y se realizará un breve análisis del funcionamiento de una aplicación tipo **App Haskell** desarrollada en este lenguaje funcional.

⥤★ ¿Qué es Haskell?

Haskell es un lenguaje de programación **funcional, puro y de tipado estático**. Fue diseñado para manejar la programación funcional de forma académica y profesional, promoviendo la claridad y corrección del código.

Es ideal para tareas como:

❤ **Desarrollo académico y de investigación**
❤ **Análisis de datos y procesamiento simbólico**
❤ **Aplicaciones concurrentes y distribuidas**
❤ **Aplicaciones web funcionales**

Se caracteriza por:

✦ Evaluación *perezosa* (lazy evaluation)
✦ Inmutabilidad
✦ Funciones como ciudadanos de primera clase
✦ Sistema de tipos poderoso (con inferencia de tipos)

◎ ¿Cómo se utiliza Haskell?

Para comenzar con Haskell, se necesita instalar el compilador **GHC (Glasgow Haskell Compiler)** y herramientas como **Stack** o **Cabal** para gestión de proyectos.

Herramientas básicas:

✿ **GHC**: Compilador oficial de Haskell
✿ **GHCi**: Intérprete interactivo
✿ **Stack**: Herramienta para crear, compilar y gestionar proyectos
✿ **Cabal**: Sistema de construcción y gestión de dependencias

```
# Crear un nuevo proyecto con Stack
stack new mi-proyecto
cd mi-proyecto
stack setup
```

```
stack build
stack exec mi-proyecto-exe
```

Ejemplo de código básico:

```haskell
-- Definición de función en Haskell
suma :: Int -> Int -> Int
suma x y = x + y
```

Haskell permite escribir código elegante, conciso y matemáticamente correcto, fomentando buenas prácticas desde su sintaxis.

⥤★ Resumen del proyecto App Haskell

La aplicación **ToDo Blog** es un ejemplo práctico del desarrollo web utilizando el lenguaje funcional **Haskell**. Este análisis examina a fondo su arquitectura, bibliotecas clave y el flujo de funcionamiento.

🔧 Tecnologías y Bibliotecas Utilizadas

✿ **Scotty**: Microframework inspirado en Sinatra que simplifica la creación de rutas y el manejo de peticiones HTTP.
✿ **Lucid**: Generador de HTML en Haskell, seguro y con tipos estáticos.
✿ **Persistent**: ORM que permite interactuar con bases de datos relacionales sin escribir SQL directamente.
✿ **SQLite**: Base de datos embebida, ideal para pruebas y prototipos.

⥤★ Estructura del Proyecto

La app se organiza de manera modular:

- **Main.hs**: Inicia la app y configura las rutas.
- **Models.hs**: Define los modelos y esquemas de la base de datos.
- **Views.hs**: Genera HTML usando funciones de Lucid.
- **Handlers.hs**: Implementa la lógica de negocio y maneja peticiones.

⥤★ Flujo de Funcionamiento

1. **Inicio del servidor**:
    - Configura Scotty para escuchar en el puerto 3000.
    - Inicializa la base de datos usando Persistent y SQLite.
2. **Rutas definidas**:
    - `GET /todos`:

        - Recupera la lista de tareas desde SQLite.
        - Usa `Lucid` para mostrar la lista en HTML.
    - `POST /todos`:
        - Recibe datos en JSON o de un formulario.
        - Guarda la nueva tarea en la base de datos.

```haskell
main :: IO ()
main = scotty 3000 $ do
  get "/todos" $ do
    todos <- liftIO getTodosFromDB
    html $ renderTodos todos

  post "/todos" $ do
    newTodo <- jsonData
    liftIO $ saveTodoToDB newTodo
    json ("Creado", newTodo)
```

🌬★ HTML Tipado con Lucid

Lucid permite escribir HTML con funciones Haskell como:

```haskell
renderTodos :: [Todo] -> Html ()
renderTodos todos = do
  html_ $ body_ $ ul_ $ forM_ todos $ \todo -> li_ (toHtml $ todoTitle todo)
```

Esto previene errores comunes de sintaxis y asegura una generación segura de contenido.

🌬★ Ventajas del enfoque funcional

✓ Mayor seguridad en tiempo de compilación
✓ Menor cantidad de errores en producción
✓ Código más fácil de razonar y testear
✓ Inmutabilidad y pureza que reducen efectos colaterales

⋆.🎋˚Conclusión

La aplicación demuestra cómo Haskell puede ser usado para construir aplicaciones web funcionales, seguras y mantenibles. Gracias a su tipado fuerte, herramientas modernas y diseño modular, es una excelente base para aprender desarrollo web funcional.
