+++ date = '2025-05-23T15:58:15-07:00' draft = false title = 'Practica 3: Aprendizaje y Análisis: Aplicación TODO en Haskell' +++

Practica 3: Analisis de Aplicacion **TODO** en Haskell

Instalacion del entorno de desarrollo

Durante la primera sesion, nos enfocamos en la instalacion del entorno necesario para trabajar con Haskell.

1. **Instalar Stack:**

Ejecutar el siguiente comando en la terminal para instalar Stack:

```
curl -sSL https://get.haskellstack.org/ | sh
```

2. **Crear un nuevo proyecto:**

Utilizar el comando `stack new` para generar la estructura básica del proyecto:

```
stack new todo
```

Este comando va a descargar una plantilla y creara una carpeta llamada `todo` con los archivos necesarios.
3. **Compilar y ejecutar el proyecto:**

Navegar al directorio del proyecto y compilarlo:

```
cd todo
stack build
```

Para ejecutar el proyecto:

```
stack run
```

Estructura del proyecto TODO en Haskell

El proyecto se organiza en varios archivos y directorios:

- `app/Main.hs:` Contiene la funcion principal `main`, que inicia la aplicacion y maneja la interaccion con el usuario.
- `src/Lib.hs:` Incluye la logica principal de la aplicacion, como las funciones para agregar, eliminar y listar tareas.

- `test/Spec.hs:` Contiene pruebas para verificar el correcto funcionamiento de las funciones definidas en `Lib.hs`.

Exploración de Haskell y Análisis de la Aplicación **TODO**

Exploramos una aplicacion de ejemplo escrita en Haskell, la cual implementa una lista de tareas (TODO). Este tipo de aplicacion sirve para gestionar tareas pendientes, con funciones basicas como agregar, eliminar y listar elementos.

Funciones principales

- **Añadir tarea:** Permite al usuario ingresar la descripción de una nueva tarea, la cual se añade a la lista de tareas pendientes.
- **Eliminar tarea:** Permite al usuario especificar una tarea (típicamente por su índice) para removerla de la lista.
- **Mostrar tarea:** Permite ver los detalles de una tarea específica, quizás con más información que en la lista general.
- **Editar tarea:** Posibilita modificar la descripción de una tarea existente o su estado (marcar como completada/pendiente).
- **Listar tareas:** Muestra todas las tareas registradas, usualmente con un índice numérico y su estado (pendiente o completada).
- **Limpiar lista**: Elimina todas las tareas de la lista, vaciando el registro.
- **Salir:** Finaliza la ejecución de la aplicación.

Funcionamiento interno

- Se hace uso de funciones de entrada/salida (IO) en Haskell, que permiten interactuar con archivos y con el usuario.
- Las tareas se manejan como listas de cadenas ([String]).
- Se usan funciones como `readFile`, `writeFile`, `putStrLn` y `getLine` para manejar datos.

Ejemplo de codigo usado en Haskell

```
import Control.Exception

import Lib (editIndex)

main :: IO ()
main = do
    putStrLn "Test:"
    let index = 1
    let new_todo = "two"
```

```
    let todos = ["Write", "a", "blog", "post"]
    let new_todos = ["Write", "two", "blog", "post"]

    let result = editIndex index new_todo todos == new_todos

    -- assert :: Bool -> a -> a
    putStrLn $ assert result "editIndex worked."
```

Ventajas de usar Haskell

- **Robustez y Confiabilidad:** El sistema de tipos estático de Haskell ayuda a prevenir muchos errores comunes en tiempo de compilación, lo que lleva a un código más robusto.
- **Claridad del Código:** La naturaleza funcional y el enfoque en funciones puras pueden hacer que el código sea más fácil de razonar y entender, ya que las funciones tienen entradas y salidas predecibles sin efectos ocultos.
- **Composición:** La capacidad de componer funciones de forma elegante permite construir funcionalidades complejas a partir de piezas más pequeñas y reutilizables.

Conclusion

Aprender Haskell me resultó desafiante debido a su naturaleza funcional. Aun así, me parecio muy interesante cómo se estructura el codigo, el uso intensivo de funciones puras y la claridad que se puede lograr en la logica.
La aplicacion TODO me ayudo a entender como manejar la entrada/salida en Haskell y como usar listas para organizar datos.