+++ date = '2025-05-22T10:18:14-08:00' draft = false title = 'Practica 3' +++

# Aplicación TODO en Haskell

Este proyecto implementa una aplicación web sencilla para gestionar tareas (una lista "TODO") utilizando Haskell. El código está estructurado de manera clara para ilustrar cómo construir una aplicación funcional básica usando el lenguaje Haskell y algunas de sus bibliotecas web.

## Estructura del Proyecto

```
examples/blog/todo/
├── Main.hs       -- Punto de entrada del servidor web
├── Todo.hs       -- Lógica principal para operaciones sobre tareas
├── Views.hs      -- Renderizado de HTML
├── Models.hs     -- Definición del modelo de datos (si aplica)
```

## Funcionamiento General

### 1. Inicio del Servidor

El archivo `Main.hs` configura e inicia un servidor web con el microframework `Scotty`. Aquí se definen las rutas que manejarán las operaciones CRUD de las tareas:

```haskell
main :: IO ()
main = scotty 3000 $ do
    get "/" showTodos
    post "/add" addTodo
    post "/update/:id" updateTodo
    post "/delete/:id" deleteTodo
```

### 2. Operaciones con Tareas

**Agregar tarea (`POST /add`)**

Se recibe el contenido de la nueva tarea desde un formulario y se almacena:

```haskell
addTodo = do
    todoText <- param "todo"
    liftIO $ addTodoToDB todoText
    redirect "/"
```

**Mostrar tareas (`GET /`)**

Obtiene todas las tareas registradas y las renderiza como HTML:

```
showTodos = do
    todos <- liftIO getTodos
    html $ renderText $ renderTodos todos
```

**Actualizar tarea (`POST /update/:id`)**

Permite editar el contenido de una tarea específica mediante su ID:

```
updateTodo = do
    id <- param "id"
    newText <- param "todo"
    liftIO $ updateTodoInDB id newText
    redirect "/"
```

**Eliminar tarea (`POST /delete/:id`)**

Borra una tarea específica, identificada por su ID:

```
deleteTodo = do
    id <- param "id"
    liftIO $ deleteTodoFromDB id
    redirect "/"
```

## Renderizado HTML (`Views.hs`)

Las vistas se generan usando la biblioteca `Lucid`, que permite construir HTML de forma declarativa en Haskell:

```
renderTodos :: [Todo] -> Html ()
renderTodos todos = html_ $ do
    head_ $ title_ "Lista de Tareas"
    body_ $ do
        h1_ "Tareas"
        ul_ $ forM_ todos $ \todo ->
            li_ (toHtml $ todoText todo)
```

## Modelo de Datos (`Models.hs`)

Define el tipo de datos para representar una tarea. También se incluyen funciones para acceder y modificar datos:

```haskell
data Todo = Todo {
    todoId :: Int,
    todoText :: Text
}

getTodos :: IO [Todo]
addTodoToDB :: Text -> IO ()
updateTodoInDB :: Int -> Text -> IO ()
deleteTodoFromDB :: Int -> IO ()
```

## Tecnologías Utilizadas

- Haskell (lenguaje de programación)
- Scotty (framework web)
- Lucid (generación de HTML)
- SQLite o almacenamiento en memoria (según la implementación)

Link de la pagina: https://lalit0o.github.io/portafolio/