# INVESTIGAR ALTERNATIVAS DEL CLIENTE

# 1. Ventajas e Inconvenientes de Elm

## Ventajas:

- **Seguridad y Robustez:** Elm es conocido por ser un lenguaje muy seguro, con un sistema de tipos que previene muchos errores comunes en tiempo de compilación. Es difícil crear código que falle en tiempo de ejecución.
- **Rendimiento:** Las aplicaciones Elm son rápidas y eficientes gracias a su optimización en la gestión del DOM virtual, similar a React, pero con un enfoque más centrado en la inmutabilidad.
- **Facilidad de Mantenimiento:** La naturaleza inmutable de Elm y su arquitectura modular facilitan el mantenimiento de proyectos a largo plazo.
- **Excelente Soporte de Errores:** Los mensajes de error del compilador de Elm son muy descriptivos y fáciles de entender, lo que facilita la depuración y el aprendizaje del lenguaje.
- **Ecosistema Consistente:** El entorno de paquetes de Elm es muy estricto, lo que garantiza que todas las bibliotecas sean compatibles entre sí, evitando problemas de dependencias.

## Inconvenientes:

- **Curva de Aprendizaje:** Aunque Elm es fácil de aprender si ya se tiene experiencia en programación funcional, puede resultar complicado para desarrolladores acostumbrados a lenguajes como JavaScript.
- **Comunidad y Ecosistema Pequeños:** Comparado con lenguajes más populares, la comunidad de Elm es pequeña, lo que puede dificultar encontrar recursos y soporte.
- **Compatibilidad:** A diferencia de JavaScript, Elm no se puede integrar fácilmente con otras bibliotecas de JavaScript, lo que limita su uso en proyectos existentes.
- **Falta de Flexibilidad:** La estricta naturaleza de Elm puede ser restrictiva para desarrolladores que prefieren más libertad en su código.

# 2. Software Necesario para Trabajar con Elm

## Entorno de Desarrollo (IDE):

- **Visual Studio Code o Atom:** Ambos son adecuados para trabajar con Elm. Existen plugins específicos para proporcionar resaltado de sintaxis, autocompletado y depuración.

## Compilador de Elm:

- **Elm Compiler:** Es necesario instalar el compilador de Elm, que convierte el código de Elm en JavaScript. Se puede instalar usando Node.js y el administrador de paquetes npm.

## Herramientas adicionales:

- **Elm REPL:** Un entorno interactivo que permite experimentar con Elm de forma inmediata, sin necesidad de crear archivos.
- **Elm Test:** Un marco de pruebas para asegurarse de que el código se comporta como se espera.
- **lm Format:** Herramienta de formato automático que asegura que el código sigue un estilo consistente.

## Pasos Básicos para la Instalación:

**1. Instalar Node.js:**

- Descargar e instalar Node.js desde la [página oficial](https://nodejs.org/).

- Verificar la instalación ejecutando node -v y npm -v en la terminal.

**2. Instalar Elm:**

- Ejecutar el comando npm install -g elm para instalar Elm de forma global.

- Verificar la instalación con elm --version.

**3. Configurar el Entorno de Desarrollo:**

- Instalar el plugin de Elm para Visual Studio Code o Atom.

- Crear un nuevo proyecto ejecutando elm init en el directorio deseado.

- Para iniciar un servidor local, usar elm reactor y acceder a http://localhost:8000 en el navegador.

# 3. Ejemplo de Código en Elm

Este ejemplo muestra una aplicación básica que maneja la entrada del usuario y actualiza un mensaje de saludo.

**Archivo: Main.elm**

```elm
module Main exposing (main)

import Browser
import Html exposing (Html, div, text, input, button)
import Html.Events exposing (onClick, onInput)
import Html.Attributes exposing (placeholder)

-- MODELO
type alias Model =
    { name : String }

initialModel : Model
initialModel =
    { name = "" }

-- MENSAJES
type Msg
    = UpdateName String
    | Reset

-- ACTUALIZACIONES
update : Msg -> Model -> Model
update msg model =
    case msg of
        UpdateName newName ->
            { model | name = newName }

        Reset ->
            initialModel

-- VISTA
view : Model -> Html Msg
view model =
    div []
        [ input [ placeholder "Introduce tu nombre", onInput UpdateName ] []
        , div [] [ text ("Hola, " ++ model.name ++ "!") ]
        , button [ onClick Reset ] [ text "Resetear" ]
        ]

-- PROGRAMA PRINCIPAL
main =
    Browser.sandbox { init = initialModel, update = update, view = view }
```

**Explicación:**

1. **Modelo (Model):** Representa el estado de la aplicación. En este caso, solo contiene una cadena (name).
2. **Mensajes (Msg):** Define los eventos que pueden ocurrir, como UpdateName y Reset.
3. **Actualización (update):** Gestiona cómo cambia el estado de la aplicación en respuesta a los mensajes.
4. **Vista (view):** Define cómo se debe renderizar la interfaz de usuario en función del estado actual.
5. **Programa Principal (main):** Configura la aplicación usando Browser.sandbox, un modo simple para aplicaciones sin efectos secundarios.