Pràctica 2 DAT

Podeu veure aquesta pràctica explicada pas per pas també al meu GitHub: https://github.com/akaKush/DAT_UPC/tree/main/P2

Primeres pàgines HTML

Fem una pàgina simple amb la URL http://soft0.upc.edu/ldatusr14/. Per fer-ho creem un arxiu anomenat **`index.html`** a la carpeta *publichtml* del nostre usuari de DAT.

El codi HTML utilitzat és el proveït per la plantilla donada a l'enunciat de la pràctica:

```
<!doctype html>
<html>
<head>
   <meta charset="UTF-8">
   <title>El títol de la vostra pàgina</title>
</head>
<body>

<h2>Títols (capçaleres) de diferents nivells (h1/h2/h3/h4/h5/
h6)</h2>

<h3>Un títol de nivell inferior</h3>

<p>Paràgrafs de text
(múltiples
línies).
<br>
Línia separada
</p>
```

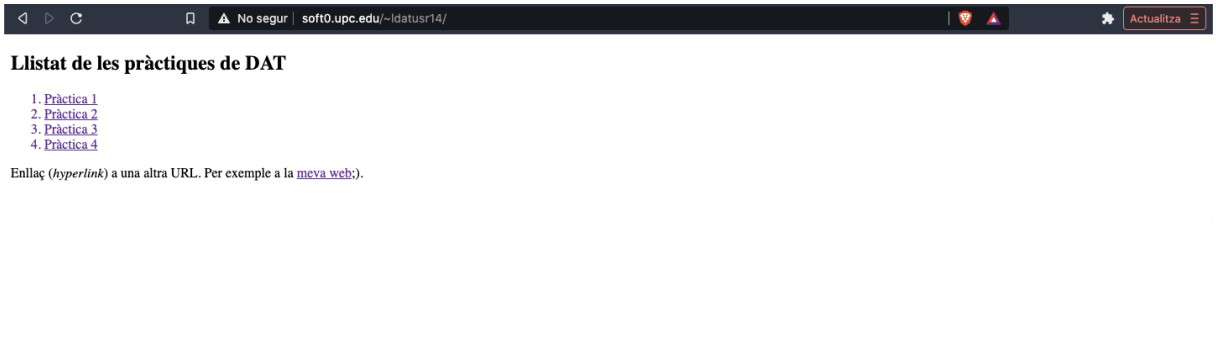
```
<p>Llista ordenada (numerada):</p>
<ol>
<li> Item 1</li>
<li> Item 2</li>
</ol>

<p>Llista no numerada:</p>
<ul>
<li> Item 1</li>
<li> Item 2</li>
</ul>

<p>Enllaç (<em>hyperlink</em>) a una altra URL. Per exemple a la
<a
href="https://akakush.github.io/marcbs/index.html">meva web</
a>.</p>

</body>
</html>
```

Tot i que el modifiquem per a que serveixi com a índex de les diferents pràctiques de l'assignatura, i ens queda així:

```
<!doctype html>
<html>
<head>
   <meta charset="UTF-8">
   <meta charset="viewport" comtent="width=device-width ,
initial-scale=1.0">
   <title>DAT - Marc Bosch</title>
</head>
<body>

<h2>Llistat de les pràctiques de DAT</h2>

<ol>
   <a href="http://soft0.upc.edu/~ldatusr14/practica1/
life-4.cgi">
      <li> Pràctica 1</li>
   </a>
```

```
    <a href="/public_html/practica2/index.html">
        <li> Pràctica 2</li>
    </a>
    <a href="# Not done yet">
        <li> Pràctica 3</li>
    </a>
    <a href="# Not done yet">
        <li> Pràctica 4</li>
    </a>
</ol>



<p>Enllaç (<em>hyperlink</em>) a una altra URL. Per exemple a la
<a
href="https://akakush.github.io/marcbs/index.html">meva web</
a>;).</p>

</body>
</html>
```

Al navegador veiem així:

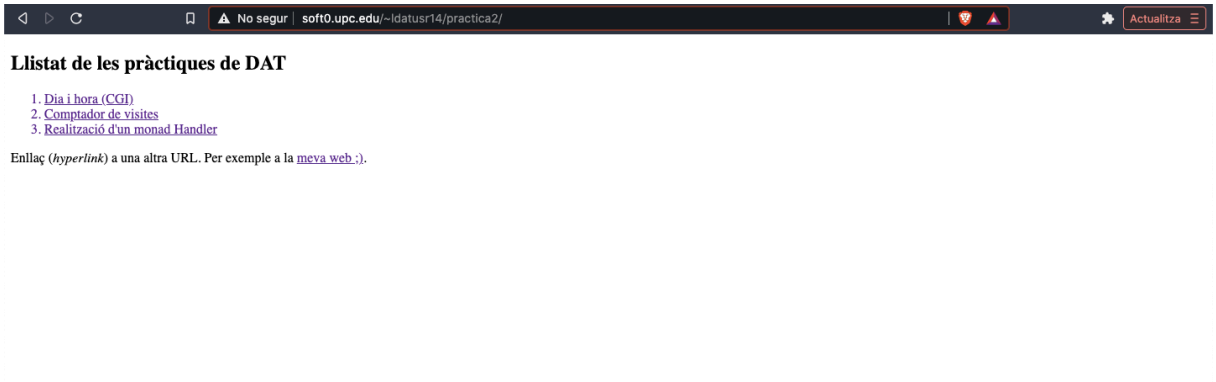


Un cop fet el índex dins la carpeta **public*html***, també creem un nou index.html dins el directori de la pràctica 2 i posem una llista de les diferents activitats que té aquesta pràctica:

```
<!doctype html>
<html>
<head>
```

```
   <meta charset="UTF-8">
   <meta charset="viewport" comtent="width=device-width ,
initial-scale=1.0">
   <title>Pràctica 2 DAT - Marc Bosch</title>
</head>
<body>

<h2>Llistat de les pràctiques de DAT</h2>

<ol>
   <a href="http://soft0.upc.edu/~ldatusr14/practica2/
dia_hora.cgi">
      <li> Dia i hora (CGI)</li>
   </a>
   <a href="http://soft0.upc.edu/~ldatusr14/practica2/
contador.html">
      <li> Comptador de visites</li>
   </a>
   <a href="http://soft0.upc.edu/~ldatusr14/practica2/calc.cgi">
      <li> Realització d'un monad Handler</li>
   </a>
</ol>



<p>Enllaç (<em>hyperlink</em>) a una altra URL. Per exemple a la
<a
href="https://akakush.github.io/marcbs/index.html">meva web ;)</
a> .</p>

</body>
</html>
```

Mirem la url al navegador:



Ara per a fer la primera part, corresponent al cgi que ens indica el dia i la hora, copiem el codi que se'ns proporciona a l'enunciat per aquest cgi, a un nou arxiu dins la carpeta de la pràctica 2 (dia*hora.cgi)*.

Per accedir a aquest cgi primer li hem de donar permisos `chmod +x dia_hora.cgi`, i llavors afegim la URL del cgi per accedir-hi desde l'index.html de la pràctica 2.

Nota: Considero que no fa falta afegir estil amb css a aquests documents html ja que la intenció final és entendre més el codi i funcionament de Haskell que el de HTML i CSS.

Comptador de visites

Per aquest apartat primer descarreguem l'arxiu del projecte que se'ns dóna a l'enunciat, i el descomprimim. Accedim a l'arxiu /src/counter.hs i l'editem perquè sigui un comptador de visites:

```
module Main
where
import System.IO
import Control.Exception

--
************************************************************

main :: IO ()
main = do
    -- Llegeix el valor del fitxer comptador i l'incrementa
    -- Escriu el nou valor al fitxer comptador
    -- Treu la sortida adeqüada (amb el nou valor)
    -- (A completar per l'estudiant)
    x <- fmap sumOne readCounter
    writeCounter
    putStrLn "Content-Type: text/plain"
    putStrLn ""
    putStrLn (show(sumOne x))

sumOne :: Int -> Int
sumOne x = x+1

readCounter :: IO Int
readCounter = do
    r <- try $ do
        h <- openFile counterFilePath ReadMode
        content <- hGetLine h
        hClose h
        return $ read content
    case (r :: Either SomeException Int) of
        Right i -> return i
        Left exc -> do
            writeCounter 0
```

```
            return 0

writeCounter :: Int -> IO ()
writeCounter i = do
    h <- openFile counterFilePath WriteMode
    hPutStrLn h $ show i
    hClose h

counterFilePath = "counter.data"
```

Fixem-nos que a l'enunciat ens demana que sigui de tipus text/plain, i per això tenim la línia "Content-Type: TIPUS*MIME*".

En el cas d'aquest exercici fem servir la funció fmap per a incrementar en 1 les visites, perquè la funció readCounter ens retorna un Int.

Un cop compilat i incrustat tot al html, veiem el següent resultat:

Nota: Important treballar amb el directori prog-web que hem descomprimit, A DINS de un directori de la practica2, el qual es troba dins de publichtml. Si no ho fem així l'script bin/make-cgi src/counter.hs NO compila ja que no troba el path correcte.

Monad Handler

En aquest exercici busquem fer servir les diverses funcions específicades a l'enunciat de la pràctica per acabar implementant una web app on trobem una calculadora de números complexes.

Per fer-ho hem de completar el fitxer *Handler.hs* en diferents part del codi que ja se'ns dóna fet.

La primera part que cal modificar és la següent:

```
instance Applicative Handler where
    -- tipus en aquesta instancia:
    --      pure  :: a -> Handler a
    --      (<*>) :: Handler (a -> b) -> Handler a -> Handler b
    pure x =
        -- (A completar per l'estudiant)
        HandlerC (\ req s0 -> pure(x,s0))
    HandlerC hf <*> HandlerC hx =
        -- (A completar per l'estudiant)
        HandlerC (\ req s0 -> do
            (f, s1) <- hf req s0
            (x, s2) <- hx req s1
            pure (f x, s2))

instance Monad Handler where
    -- tipus en aquesta instancia:
    --      (>>=) :: Handler a -> (a -> Handler b) -> Handler b
    HandlerC hx >>= f =
        -- (A completar per l'estudiant)
        HandlerC $ \ req s0 -> do
            (x, s1) <- hx req s0
            let HandlerC hy = f x
            hy req s1
```

Unes línies més avall modifiquem el següent:

```
-- Obte informació de l'estat del handler
getsHandlerState :: (HandlerState -> a) -> Handler a
getsHandlerState f =
    -- (A completar per l'estudiant)
    HandlerC (\ req s0 -> pure (f s0, s0))

-- Modifica l'estat del handler
modifyHandlerState :: (HandlerState -> HandlerState) -> Handler
()
modifyHandlerState f =
```

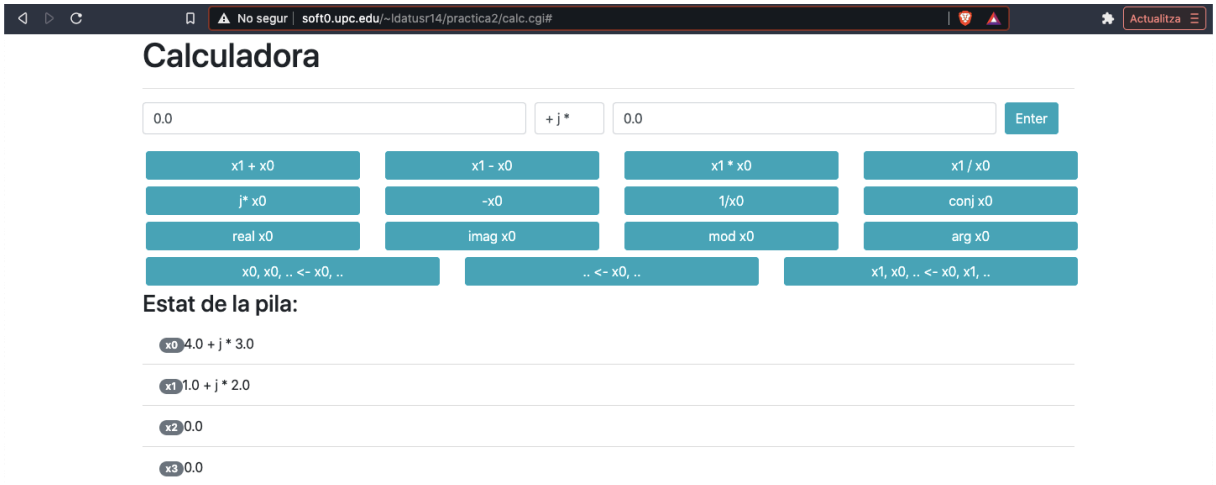
```
    -- (A completar per l'estudiant)
    HandlerC (\ req s0 -> return ((), f s0))
```

Finalment trobem l'últim tall a modificar:

```
-- Obte els valors associats al parametre de la peticio amb el
nom indicat.
lookupPostParams :: Text -> Handler [Text]
lookupPostParams name = do
    -- Monad Handler:
    mbparams <- postParams
    case mbparams of
        Just params -> -- params es una llista de parelles de
tipus (Text, Text)
            -- (A completar per l'estudiant).
            -- Caldra obtenir tots els valors (segon component)
de les parelles que tenen el nom (primer component) igual al
indicat.
            -- NOTA: Useu les funcions
            --   fst :: (a, b) -> a
            --   snd :: (a, b) -> b
            --   filter :: (a -> Bool) -> [a] -> [a]
            return (map snd (filter (\ param -> fst param ==
name) params))
        Nothing ->
            -- El contingut de la peticio no es un formulari. No
hi ha valors.
            pure []
```

Un cop editat tot el fitxer, el guardem i compilem amb `bin/make-cgi src/calc.hs` situats dins el directori `public_html/practica2/prog-web`.

Per executar la calculadora i veure el seu funcionament ho podem fer al següent link: http://soft0.upc.edu/


No segur | soft0.upc.edu/~ldatusr14/practica2/calc.cgi#
Actualitza
Calculadora
0.0
+ j *
0.0
Enter
x1 + x0
x1 - x0
x1 * x0
x1 / x0
j* x0
-x0
1/x0
conj x0
real x0
imag x0
mod x0
arg x0
x0, x0, .. <- x0, ..
.. <- x0, ..
x1, x0, .. <- x0, x1, ..
Estat de la pila:
x0 4.0 + j * 3.0
x1 1.0 + j * 2.0
x2 0.0
x3 0.0