Programación (en Java) para apoyar la aceptación universal

Programa de Capacitación para la Aceptación Universal de LACRALO
Una colaboración LACRALO-UASG-ICANN

Sarmad Hussain y Farah Adeeba

18 de mayo de 2021

ICANN

LACRALO ICANN

Universal Acceptance

Programa de Capacitación sobre UA de LACRALO

Descripción general

Fecha	Sesión	Destinatarios	Descripción
4 de mayo de 2021	Introducción a la UA	General	Una introducción a los aspectos fundamentales de la UA y la EAI
11 de mayo de 2021	Configuración de la EAI	Técnicos (administradores de correo electrónico y sistemas)	Una capacitación detallada sobre cómo configurar los sistemas de correo electrónico para que sean compatibles con EAI.
18 de mayo de 2021	UA para desarrolladores de Java	Técnicos (desarrolladores de software)	Una capacitación detallada sobre cómo diseñar y desarrollar aplicaciones y sistemas compatibles con la Aceptación Universal.
25 de mayo de 2021	Cómo participar en las actividades de Aceptación Universal	General	Una sesión para debatir cómo los participantes pueden seguir involucrados en las iniciativas de UA en toda la región de LAC.

- Se entregarán certificados a los participantes que asistan a las sesiones y completen las encuestas de cada una de las 4 sesiones..

Hola!

Dr. Sarmad Hussain
Sr. Director IDN & UA Programs
ICANN
sarmad.Hussain@icann.org

Dr. Farah Adeeba
Assistant Professor
KICS University of Engineering and Technology
farah.adeeba@kicks.edu.pk

Agenda

- Introducción (5 minutos)
- Visión general de la aceptación universal (15 minutos)
  - Quiz (5 minutos)
- Fundamentos para los IDN y el EAI (15 minutos)
  - Quiz (5 minutes)
- Programación para UA usando Java (35 minutos)
  - Procesamiento de nombres de dominio
  - Procesamiento de la dirección de correo electrónico
  - Almacenamiento de nombres de dominio y direcciones de correo electrónico
  - Nombres de dominio y direcciones de correo electrónico en la plataforma Android
- Conclusión (10 minutos)

Visión general de la Aceptación Universal

Objetivo

Que todos los nombres de dominio y direcciones de correo electrónico funcionen en todas las aplicaciones de software.

Impacto

Promover la elección de los consumidores, mejorar la competencia y proporcionar un acceso más amplio a los usuarios finales.

Categorías de nombres de dominio y direcciones de correo electrónico

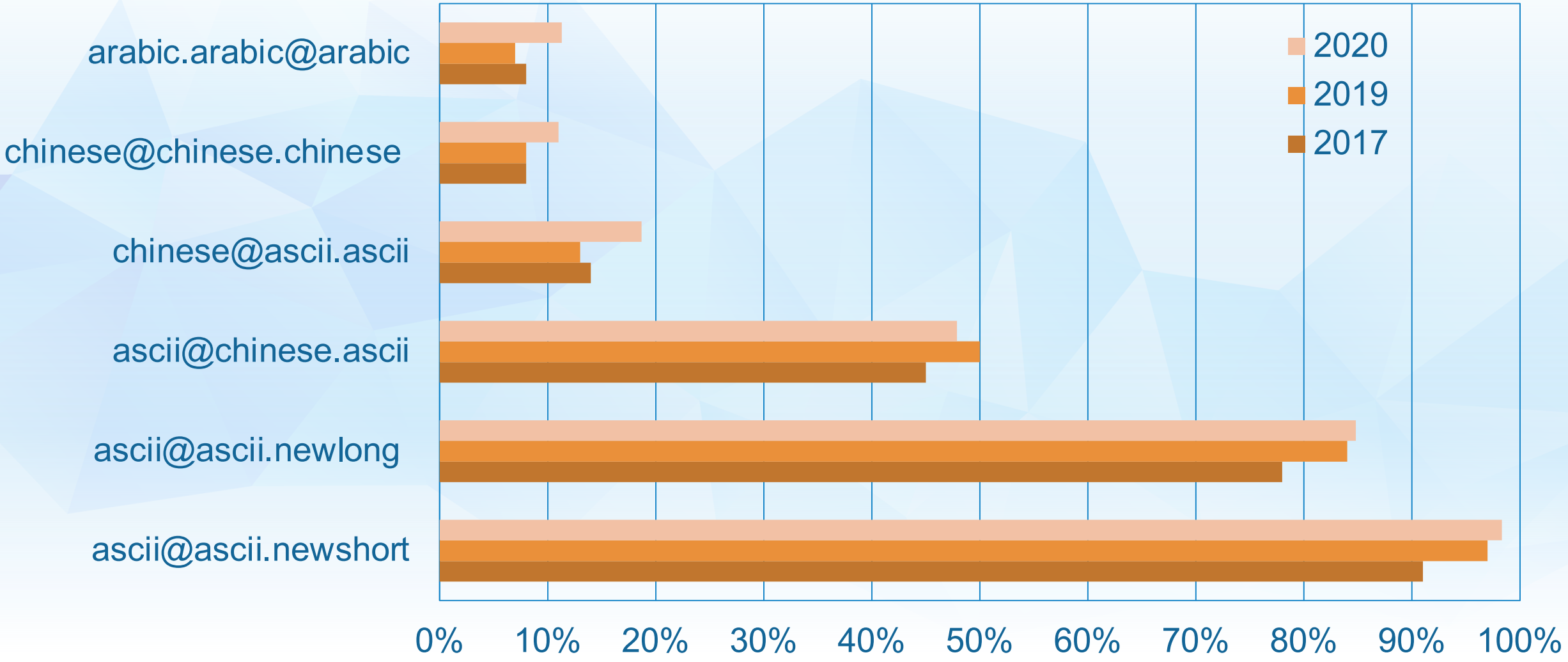
- Ahora es posible tener nombres de dominio y direcciones de correo electrónico en los idiomas locales.
  - Nombres de dominio internacionalizados (IDN)
  - Internacionalización de direcciones de correo electrónico (EAI)
  - Formato UTF8 de Unicode utilizado para IDN y EAI

- Nombres de Dominio
  - **Nuevos** nombres de dominio de nivel superior: example.sky
  - Nombres de dominio de nivel superior **más largos**: example.abudhabi
  - Nombres de dominio **internacionalizados** 普遍接受-测试.世界

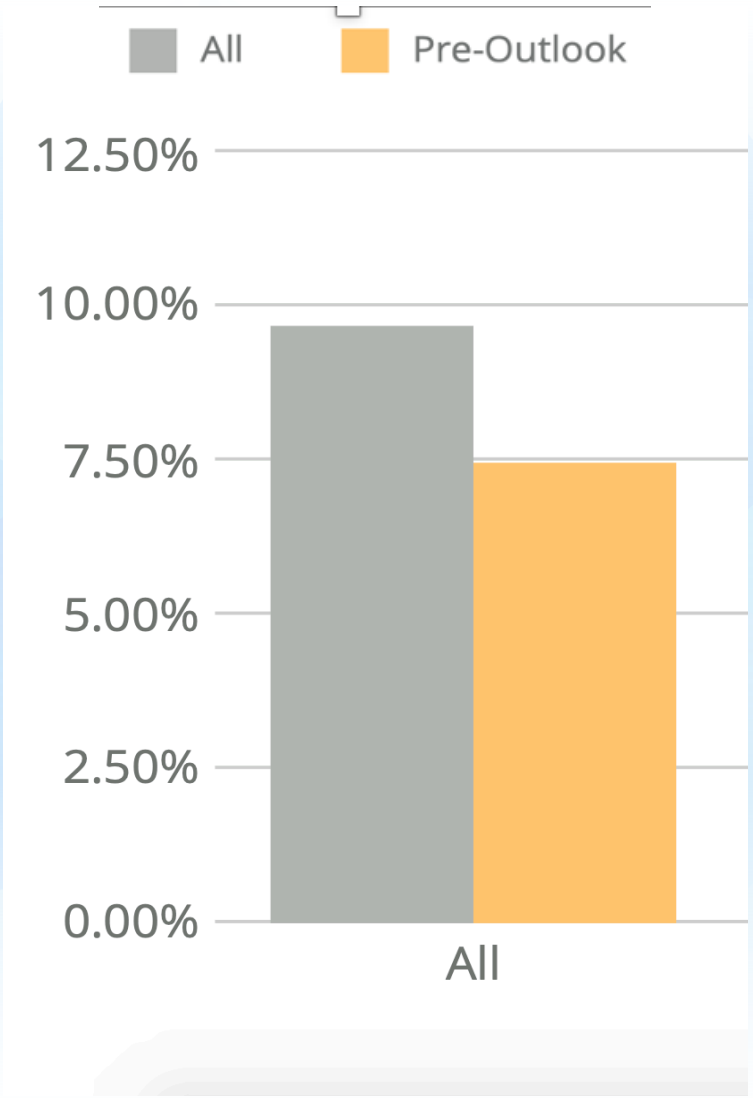
- Direcciones de correo electrónico internacionalizadas (EAI)
  - ASCII@IDN marc@société.org
  - UTF8@ASCII ईमेल@example.com
  - UTF8@IDN 测试@普遍接受-测试.世界
  - UTF8@IDN; right to left scripts ای-میل@مثال.موقع

Aceptación de direcciones de correo electrónico en sitios web a nivel mundial

Para obtener más detalles, consulte UASG027



Soporte estimado de EAI en sistemas de correo electrónico para todos los TLD



Solo el 9,7% de los dominios muestreados estaban listos para EAI en 2019

basado en servidores de correo encontrados a través de registros MX en zonas de todos los TLD

Para obtener detalles sobre la metodología, consulte
UASG021D: EAI Readiness in TLDs

Alcance de la preparación de UA para programadores

1. Apoyar todos los nombres de dominio

2. Soporte de todas las direcciones de correo electrónico

Pila (Stack) de tecnología para la consideración de UA

Aplicaciones y sitios web

- Wikipedia.org, ICANN.org, Amazon.com, sitios web personalizados a nivel mundial
- PowerPoint, Google-Docs, Safari, Acrobat, aplicaciones personalizadas

Redes sociales y motores de búsqueda

- Chrome, Bing, Safari, Firefox, local (e.g., Chinese) browsers
- Facebook, Instagram, Twitter, Skype, WeChat, WhatsApp, Viber

Lenguajes y marcos de programación

- JavaScript, Java, Swift, C#, PHP, Python
- Angular, Spring, .NET core, J2EE, WordPress, SAP, Oracle

Plataformas, sistemas operativos y herramientas del sistema

- iOS, Windows, Linux, Android, App Stores
- Active Directory, OpenLDAP, OpenSSL, Ping, Telnet

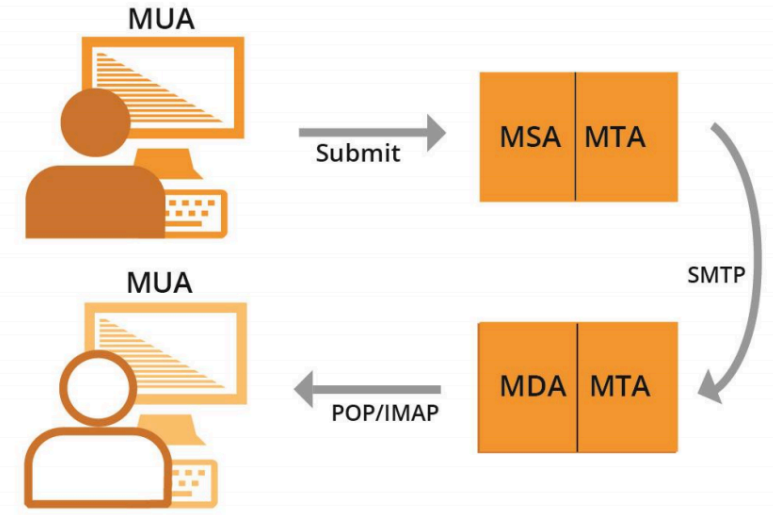
Estándares y mejores prácticas

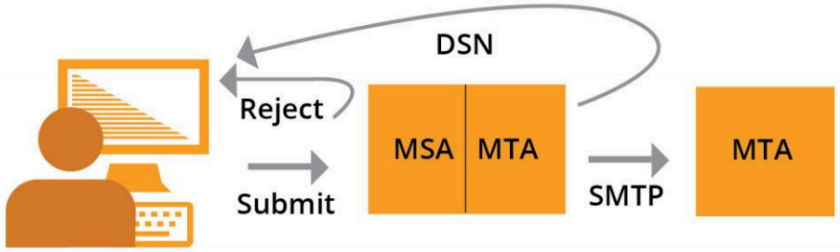
- IETF RFCs, W3C HTML, Unicode CLDR, WHATWG
- Normas basadas en la industria (salud, aviación, ...)

Aceptar, validar, procesar, almacenar y mostrar todos los nombres de dominio y direcciones de correo electrónico.

Sistemas de correo electrónico y soporte EAI

- Todos los agentes de correo electrónico deben estar configurados para enviar y recibir direcciones de correo electrónico internacionalizadas. Ver EAI: A Technical Overview para más detalles.
  - MUA – Mail User Agent: Un programa cliente que una persona utiliza para enviar, recibir y administrar correo.
  - MSA – Mail Submission Agent: Un programa de servidor que recibe correo de un MUA y lo prepara para la transmisión y entrega.
  - MTA – Mail Transmission Agent: Un programa de servidor que envía y recibe correo desde y hacia otros hosts de Internet. Una MTA puede recibir correo de un MSA y/o entregar correo a un MDA.
  - MDA – Mail Delivery Agent: Un programa de servidor que controla el correo entrante y normalmente lo almacena en un buzón o carpeta.





Quiz

Quiz – Pregunta 1

- Para mejorar los sistemas que están listos para la Aceptación Universal (UA), ¿cuáles de las siguientes categorías de nombres de dominio y direcciones de correo electrónico son relevantes?
  1. Nombres de dominio ASCII.
  2. Nombres de dominio internacionalizados (IDNs).
  3. Direcciones de correo electrónico internacionalizadas (EAI).
  4. Todo lo anterior.
  5. Sólo 2 y 3.

Fundamentos para nombres de dominio y direcciones de correo electrónico internacionalizados

Codificación Unicode

- Unicode codifica glifos en puntos de código para diferentes scripts del mundo.
  - Codepoints que se muestran en hexadecimos mediante la notación U+XXXX.
  - Archivos Unicode normalmente en formato UTF8, utilizando un número variable de bytes para un punto de código.
  - ASCII se utiliza tal cual en Unicode: e = ASCII 65 = U+0065.
- Hay múltiples maneras de codificar ciertos glifos en Unicode:
  - è = U+00E8
  - e + ` = è = U+0065 + U+0300
- La normalización garantiza que la representación final sea la misma, incluso si los usuarios escriben de forma diferente.
  - Los estándares IDN recomiendan usar Normalization Form C (NFC).
  - Genera U+00E8 para ambas versiones de entrada anteriores.

Nombres de dominio y nombres de dominio internacionalizados (IDNs)

- Un nombre de dominio es un conjunto ordenado de etiquetas o cadenas: www.example.co.uk.
  - El dominio de nivel superior (TLD) es la etiqueta más a la derecha: ”uk”
  - Inicialmente, los TLD tenían sólo dos o tres caracteres de largo (p. ej., .ca, .com).
  - Ahora los TLDs pueden ser cadenas más largas (p. ej.., .info, .google, .engineering).
  - Los TLD delegados en la zona raíz pueden cambiar con el tiempo, por lo que una lista fija puede quedar desactualizada.
- Los nombres de dominio también se pueden internacionalizar cuando una de las etiquetas contiene al menos un carácter no ASCII.
  - por ejemplo: www.exâmple.ca o 普遍接受-测试.世界.
- Utilice el último estándar IDN llamado IDNA2008 para IDNs.
  - No utilice bibliotecas para la versión obsoleta IDNA2003.

IDN y EAI

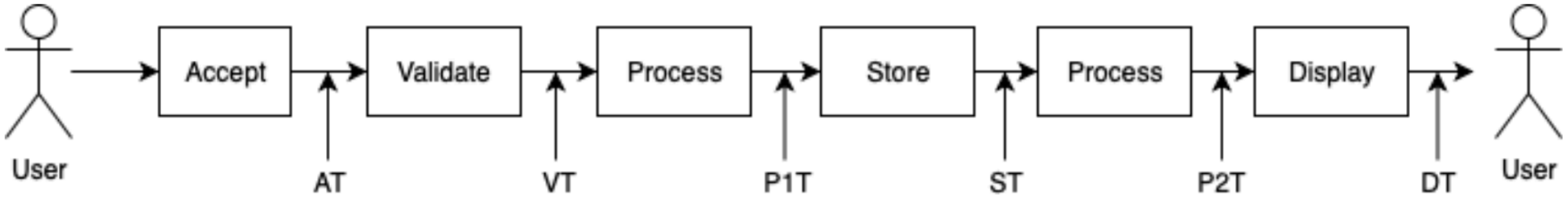
- Hay dos formas equivalentes de etiquetas de dominio IDN: etiqueta U y etiqueta A.
  - Los usuarios humanos utilizan la versión IDN llamada etiqueta U (usando el formato UTF-8): exâmple
  - Las aplicaciones o sistemas utilizan internamente un equivalente ASCII llamado etiqueta A:
    1. Tome la entrada del usuario y normalice y compruebe con IDNA2008 para formar idn U-label.
    2. Convierta etiqueta U en punycode (usando RFC3492).
    3. Agregue el prefijo "xn--" se agrega para identificar la cadena ASCII como una etiqueta A IDN.
       - exâmple => exmple-xta => xn--exmple-xta
       - 普遍接受-测试 => --f38am99bqvcd5liy1cxsg => xn----f38am99bqvcd5liy1cxsg
- Sintaxis de dirección de correo electrónico: mailboxName@domainName
  - EAI tiene el nombre del buzón en Unicode (en formato UTF8).
  - El domainName puede ser ASCII o IDN.
    - por ejemplo: kévin@example.org o すし@快手.游戏.

Cuestiones de Aceptación Universal

- Algunas aplicaciones siguen verificando incorrectamente los nombres de dominio mediante uno de los métodos obsoletos:
  - Compruebe si hay una longitud fija de TLD entre 2-4 caracteres (TLD puede tener hasta 63 caracteres).
  - Compruebe desde un conjunto fijo de TLDs, por ejemplo, utilizando la lista estática de cadenas.
  - Compruebe si solo hay caracteres ASCII.
- Algunas aplicaciones no cumplen con los requisitos adicionales para validar el EAI:
  - Compruebe que el nombre del buzón es una cadena válida en formato UTF-8.
  - El nombre de dominio puede ser ASCII o IDN.

Marco de pruebas de preparación de UA

- Basado en UASG026, los componentes de la aplicación pueden generalizarse para poner énfasis en el procesamiento de identificadores internacionalizados.
- Cada puerta tiene su propio conjunto de requisitos y procesamiento.



- AT: Aceptar prueba
- VT: Validar la prueba
- P1T: Prueba de proceso en la entrada
- ST: Prueba de almacenamiento
- P2T: Prueba de proceso en la salida
- DT: Prueba de visualización

Validación de la entrada del usuario

- Validar la entrada del usuario, o cualquier entrada, es extremadamente útil por varias razones, algunas de las cuales incluyen: una mejor experiencia de usuario, mayor seguridad y evitar problemas irrelevantes.

- Validar nombres de dominio y direcciones de correo electrónico es útil.

- Algunos métodos de validación para nombres de dominio y direcciones de correo electrónico:
  - Comprobaciones de sintaxis básicas: ¿es correcta la sintaxis de la cadena?
    - ¿Contiene el nombre de dominio '.' ?
    - ¿La dirección de correo electrónico contiene '@' y una parte válida del nombre de dominio?
  - Comprobaciones funcionales: funciona el nombre de dominio o la dirección de correo electrónico?
    - ¿Está en uso el dominio de nivel superior (TLD)?
    - ¿Está todo el nombre de dominio en uso?
    - ¿Está en uso el correo electrónico?

Validación del nombre de dominio

- Validación de la sintaxis:
  - ASCII: RFC1035
    - Compuesto de letras, dígitos y guión.
    - La longitud máxima es de 255 octetos con cada etiqueta de hasta 63 octetos.
  - IDN: IDNA2008 (RFCs 5890-5894)
    - Etiquetas A válidas
    - Etiquetas U válidas

- Función de validación:
  - Es el dominio de nivel superior (TLD) en uso?
    - Verificar contra la lista de TLDs.
    - Verifique mediante una solicitud DNS.
  - ¿Está todo el nombre de dominio en uso?
    - Verifique mediante una solicitud DNS.

Resolución del nombre de dominio

- Después de la validación, un software usaría el identificador de nombre de dominio como:
  - Un nombre de dominio que se resolverá en el DNS.
- Por lo tanto, para ser compatible con UA, el software tiene que utilizar métodos adecuados que admitan UA.
  - Por ejemplo, es posible que pasar una etiqueta U a las funciones o métodos tradicionales no se realice correctamente, ya que no espera un nombre de dominio UTF8.

Validación de la dirección de correo electrónico

- Una dirección de correo electrónico se compone de: mailboxName@domainName
- Validación de la sintaxis:
  - Para domainName, vea la discusión anterior.
  - Para mailboxName:
    - ASCII
    - UTF8 (para el EAI)
- Función de validación:
  - ¿Está configurado el nombre de dominio para enviar y recibir correos electrónicos?
  - ¿Es el nombre del buzón capaz de enviar y recibir correos electrónicos?

Envío de correo electrónico

- Después de la validación, un software usaría el identificador de correo electrónico como:
  - Un id de usuario basado en direcciones de correo electrónico.
  - Para enviar un correo electrónico.
- Por lo tanto, para ser compatible con UA, el software debe utilizar métodos adecuados que admitan UA.
  - Por ejemplo, es posible que pasar una dirección de correo electrónico de nombre de buzón UTF8 a un remitente de correo no se realice correctamente, ya que no espera un nombre de buzón UTF8 en la dirección de correo electrónico.

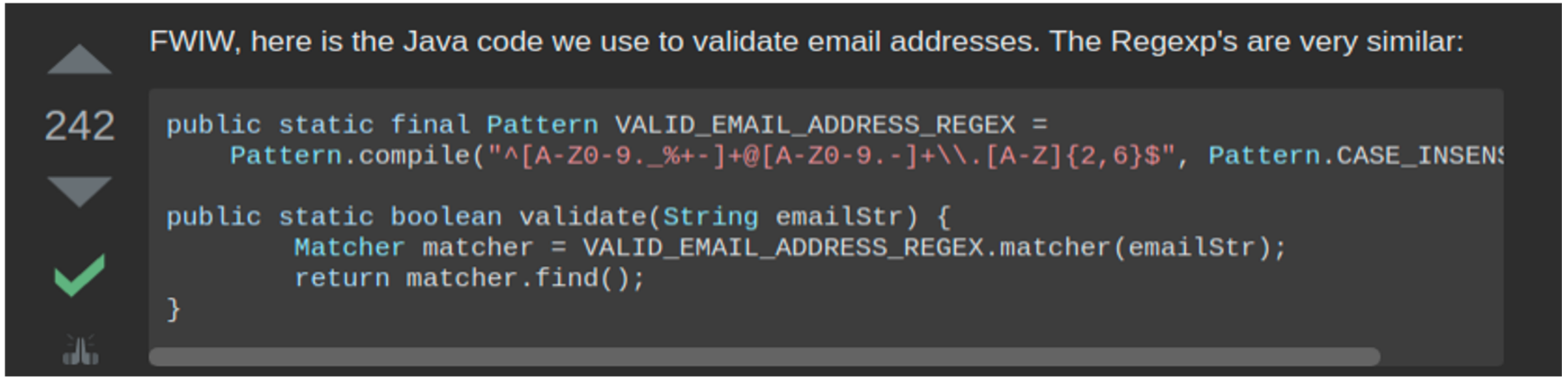
Envío de correo electrónico

- Se documenta una lista completa de casos de pruebas de UA en UASG004.
- Se recomienda encarecidamente a los desarrolladores que utilicen estos casos de prueba en su unidad y pruebas del sistema.

Quiz

Quiz 2: Un ejemplo real

- Una empresa construyó un sitio web donde los consumidores internacionales pueden suscribirse a través de su correo electrónico. Dado que el formulario de suscripción es de entrada de usuario, los desarrolladores validaron la dirección de correo electrónico antes de intentar enviar el correo electrónico.
  - Los desarrolladores fueron a Stackoverflow y encontraron una expresión regular (regex) para realizar la validación:



  - El regex limita el buzón a las letras A-Z, dígitos 0-9 y algunos símbolos, las etiquetas de dominio a letras, dígitos y guión, y el dominio de nivel superior a letras solo con longitud 2-6.

- ¿Funcionaría este regex para el sitio web de la empresa? ¿Por qué o por qué no?

Programación para UA usando Java

Versión Java

- Los ejemplos de código se probaron en Java 11 (versión de Oracle) y Android API/SDK 26 cuando corresponda.
  - Es posible que las versiones anteriores tengan problemas.
- Algunas bibliotecas pueden requerir (no necesariamente debido a UA) versiones más recientes de Java.
- A menos que se indique explícitamente, este tutorial debe aplicarse a cualquier tipo de máquina virtual:
  - Oracle, OpenJDK, o Android (Dalvik/ART).

Versiones de la biblioteca Java

- En este tutorial se muestran muchas bibliotecas como se encuentra en el campo. Si bien la lista no es exhaustiva, sigue siendo completa para ayudarle a evaluar cuál y cómo utilizar una biblioteca, especialmente si el software ya ha sido desarrollado.
  - El informe de la UASG proporciona información detallada sobre las bibliotecas que cumplen con UA. ver https://uasg.tech/wp-content/uploads/documents/UASG018A-en-digital.pdf.

- Las bibliotecas que se muestran en este tutorial se han probado en la versión actual disponible en el momento de escribir.
  - Es posible que las nuevas versiones de algunas bibliotecas hayan solucionado problemas o realizado mejoras que cambien las recomendaciones.
  - Por favor, compruebe en el momento de su desarrollo el estado de estas bibliotecas.

Titular de tipo para identificadores de UA

- Los identificadores de UA son nombres de dominio y direcciones de correo electrónico.
  - Estos identificadores pueden contener datos Unicode UTF8.
- Java String type es adecuado para contener esos identificadores, ya que es compatible de forma nativa con Unicode. Por lo tanto, la mayoría de las bibliotecas esperan el tipo String.
- El conjunto de caracteres predeterminado (Charset.defaultCharset()) suele ser UTF8 en la mayoría de los sistemas. Verifique (java -XshowSettings) o cambie el conjunto de caracteres predeterminado en la máquina virtual Java que está utilizando.
  - Para obtener más información, vea JEP.

Algunos casos básicos de prueba

- Los ejemplos de código a lo largo del tutorial utilizarán estas entradas, que proporcionan casos básicos de prueba de UA (no exhaustivos):

```
List<String> testDomains = List.of(
   "example.org",                  // ascii.ascii
   "example.undefinedtld",         // unexistant tld
   "example.recentTld",            // recently allocated tld
   "example.accountants",          // allocated longer than 7 char tld
   "exâmple.org",                  // ulabel.ascii
   "xn--exmple-xta.org",           // alabel.ascii
   "exâmple.ไทย",                  // ulabel.ulabel
   "exâmple.xn--o3cw4h",           // ulabel.alabel
   "xn--exmple-xta.xn--o3cw4h"     // alabel.alabel
 );
 List<String> testLocalParts = List.of(
   "user",
   "kévin"
 );
```

Procesamiento de nombres de dominio

Versiones de bibliotecas Java

- Forma tradicional de hacer resolución de nombres de host y resolución de sockets:

```
import java.net.InetAddress;
getByName(String host);
getAllByName(String host);
Socket(String host, int port);
```

  - Utiliza getByName()

- Produce un UnknownHostException para cualquier error:
  - No se devuelven direcciones IP
  - Host no válido
  - Mala sintaxis
  - ...

- Pasa el host String tal cual al sistema operativo subyacente, sin validación.
  - Por lo tanto, se pasan dominios no válidos (como etiquetas U no válidas).
  - Depende de la implementación del sistema operativo subyacente.

Uso de Java simple: Recomendación

- No utilice as-is con un nombre de host. En lugar de:
  - Prepare el nombre de host (por ejemplo, convertir IDN U-label para A-label) utilizando otra biblioteca y luego usar estas llamadas base.
  - Valide el nombre de host antes de llamar getByName()
    - Para evitar retrasos a la espera de respuestas de las consultas que se lanzarán de todos modos.
    - Proporcione mejores comentarios al usuario porque el lanzamiento no le indicará si el nombre de host estaba incorrecto o si el nombre de host estaba bien, pero la consulta no devolvió datos.

Componentes internacionales para Unicode (UCI)

- La biblioteca estándar gold para Unicode. Fue desarrollado por IBM y ahora está gestionado por Unicode. En sincronía con los estándares Unicode.
  - ICU4J (http://site.icu-project.org/home) no es realmente Java-ized. Es una asignación directa a la versión C. Es posible que a los desarrolladores de Java no les guste por esa razón.
  - Conversión IDNA se basa en Unicode UTS46, que soporta la transición de IDNA2003 a IDNA2008. Sin embargo, es posible configurar para no admitir la transición (recomendado).
  - Conversión IDNA incluye normalización según IDNA (¡bueno!).
  - La salida de los métodos puede contener nombres de dominio incorrectos, ya que los caracteres no permitidos se reemplazan por U+FFFD.
  - Compruebe si hay errores en la conversión llamando a info.hasErrors().
  - Para los IDN, establezca las opciones para restringir la validación y utilizarla en IDNA2008.
  - Los métodos estáticos implementan IDNA2003 y los métodos no estáticos implementan IDNA2008.

Uso de la UCI

- Repositorio maven:

```
<dependency>
 <groupId>com.ibm.icu</groupId>
 <artifactId>icu4j</artifactId>
 <version>65.1</version>
</dependency>
```

```
import com.ibm.icu.text.IDNA;
IDNA validator = IDNA.getUTS46Instance(
    IDNA.NONTRANSITIONAL_TO_ASCII
  | IDNA.NONTRANSITIONAL_TO_UNICODE
  | IDNA.CHECK_BIDI
  | IDNA.CHECK_CONTEXTJ
  | IDNA.CHECK_CONTEXTO
  | IDNA.USE_STD3_RULES);
StringBuilder output = new StringBuilder();
IDNA.Info info = new IDNA.Info();
validator.nameToASCII(input, output, info);
if (info.hasErrors()) {}
```

Opción de no usar UTS46 función de transición y para mejorar la validación.

Xcode

- Biblioteca de utilidades desarrollada por Verisign. Tiene un objeto "Idna".
  - https://www.verisign.com/en_US/channel-resources/domain-registry-products/idn-sdks/index.xhtml
  - No Hay repositorio Maven (sólo un zip descargable con un archivo jar en él).
  - Lento (las pruebas muestran que el procesamiento de un dominio puede tardar hasta 5 segundos) pero implementa IDNA2008 perfectamente.

```
import com.vgrs.xcode.common.Unicode;
import com.vgrs.xcode.idna.Idna;
import com.vgrs.xcode.idna.Punycode;
 Idna idna = new Idna(new Punycode(), true, true);
 int[] output = idna.domainToUnicode(input.toCharArray());
        // see domainToAscii for roundtrip
 String domain = new String(Unicode.decode(output));
```

Otras bibliotecas

- JRE-IDN
  - Incluido en JRE pero implementa IDNA2003.
- Validador Común Apache
  - Tiene validadores de dominio y correo electrónico.
  - ¡No lo use, ya que se basa en una lista estática de TLDs! ANTICUADO!
- Guava
  - Biblioteca de utilidades desarrollada por Google.
  - No es útil para la validación.
  - Si se usa, tenga en cuenta que depende de la lista de sufijos públicos, establecida estáticamente en la biblioteca, por lo que tendría que actualizar la biblioteca con frecuencia.

Hacer una solicitud HTTP

Cliente HTTP	Aceptar Formulario UTF8	Normalización	Conversión automática	IDNA2008
Java 1.1 HttpURLConnection	✗	✗	✗	✗
Apache HTTP Client	✗	✗	✗	✗
OkHTTP	✓	✓	✓	✗
Java 11 HTTP Client	✗	✗	✗	✗
Google Java HTTP Client	✗	✗	✗	✗

Procesamiento de direcciones de correo electrónico

Validación del correo electrónico

Expresiones regulares de correo electrónico (Regex)

- Básico: something@something
  - ^(.+)@(.+)$
- De owasp.org (seguridad):
  - [^[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,7}$]
    - No admite EAI, es decir, nombre de buzón de correo en UTF8 no permitido: [a-zA-Z0-9_+&*-]
    - No admite ASCII TLD de más de 7 caracteres: [a-zA-Z]{2,7}
    - No admite etiquetas U en IDN TLD: [a-zA-Z]
  - Pero OWASP es la referencia para la seguridad.
    - Por lo tanto, puede terminar luchando con su equipo de seguridad para utilizar un Regex compatible con UA en lugar del "estándar" de OWASP.

Expresiones regulares de correo electrónico (Regex)

- Ejemplo de Regex sugerido en varios foros: ex: List of proposals
  - ^[A-Za-z0-9+_.-]+@(.+)$ no es compatible con UTF8 en nombre de buzón
  - ^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$ no es compatible con las etiquetas en U
  - ^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+(?:\\.[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+)*@[a-zA-Z0-9-]+(?:\\.[a-zA-Z0-9-]+)*$ no es compatible con las etiquetas en U
  - ^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$ tienen restricciones de longitud para el TLD entre 2 – 6 caracteres

- Uno puede llegar a un regex compatible con EAI-IDN utilizando varias características de puntos de código Unicode.
  - Para IDN sería como una reimplementación de las tablas de protocolo IDNA en regex!

- Dado que ambos lados de un EAI pueden tener UTF8, entonces un regex para un EAI podría ser .*@.* que sólo está verificando la presencia del carácter '@'.

Jakarta Mail

- Paquete Java más utilizado para enviar correo electrónico; también tiene un método validate() para direcciones de correo electrónico.
- valid() hace un buen trabajo en la validación de direcciones de correo electrónico, especialmente la parte local:
  - Verifica caracteres ilegales como: ()<>,;:"[]\ , espacios en blanco, etc.
  - Comprueba que los caracteres son solo dígitos y letras según la definición de clases Unicode.
  - No valida el IDN, así que agregue la validación y preparación del IDN como paso adicional.

Uso de Jakarta Mail

- `import javax.mail`

- Maven:

```
<dependency>
  <groupId>com.sun.mail</groupId>
  <artifactId>jakarta.mail</artifactId>
  <version>1.6.5</version>
</dependency>
```

```
import javax.mail.internet.InternetAddress;
try {
 InternetAddress emailAddr = new InternetAddress(input);
 emailAddr.validate();
} catch (AddressException e) {}
```

Otras bibliotecas

- Apache Commons Validator
  - Tiene validadores de dominio y correo electrónico.
  - ¡No lo use, ya que se basa en una lista estática de TLDs! ANTICUADO!
- EmailValidator4J
  - ¡Reclamaciones de apoyo al EAI! (no probado)
  - https://github.com/egulias/EmailValidator4J

Envío de correo electrónico

Uso de Jakarta Mail para enviar correo electrónico

- La misma biblioteca y Maven que antes.

```
Properties properties = System.getProperties();
      properties.setProperty("mail.smtp.host", host);
      Session session = Session.getDefaultInstance(properties);
      try {
        MimeMessage message = new MimeMessage(session);
        message.setFrom(new InternetAddress(from));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
        message.setSubject("This is the Subject Line!");
        message.setText("This is actual message");
        Transport.send(message);
      } catch (MessagingException e) {
      }
```

Java Mail Simple

- Jakarta Mail wrapper; llamadas más simples para enviar correos electrónicos.
  - Utiliza (e incluye) otra biblioteca para la validación de correo electrónico.
  - Utiliza varios Regex.
  - Considera que cualquier UTF8 no es válido, por lo tanto, no admite la etiqueta U como dominio o EAI.
  - Validación interna basada en el RFC2822 obsoleto.

Un ejemplo real

- El equipo de desarrollo se dio cuenta de que el paquete com.sun.mail:javax.mail:1.5.6 utilizado para enviar confirmación de suscripción por correo electrónico a través de SMTP ya tenía una función de "validación". Reescriben el método isEmailValid:

```
public static boolean isEmailValid(String email) {
  try {
    var iEmail = new javax.mail.internet.InternetAddress(email);
    iEmail.validate();
    return true;
  } catch (AddressException e) {
    return false;
  }
}
```

- Sin embargo, encontraron que el método sigue fallando.

Un ejemplo real

- Encontraron que esta característica de internacionalización se corrigió en una versión más reciente, por lo que actualizaron a: com.sun.mail:jakarta.mail v1.6.5

```
public static boolean isEmailValid(String email) {
  try {
    var iEmail = new javax.mail.internet.InternetAddress(email);
    iEmail.validate();
    return true;
  } catch (AddressException e) {
    return false;
  }
}
```

- Por último, al inspeccionar estas correcciones en javamail y su versión renombrada jakartamail, se dieron cuenta de que necesitaban también modificar la función de suscripción y su servidor SMTP para admitir una nueva marca "SMTPUTF8".

Un ejemplo real

- El equipo de desarrollo de la compañía hizo parte del trabajo. Los correos electrónicos que utilicen IDN seguirán siendo rechazados por Jakarta Mail.

- Aquí está el ejemplo completo, preparación de la dirección de correo electrónico con etiquetas A en el dominio, que luego se utiliza como entrada a cualquier biblioteca o marco de trabajo.

- La parte local sigue siendo UTF8 que puede no estar trabajando en la ruta de entrega de correo.

```
IDNA validator = IDNA.getUTS46Instance(
    IDNA.NONTRANSITIONAL_TO_ASCII
    | IDNA.NONTRANSITIONAL_TO_UNICODE
    | IDNA.CHECK_BIDI
    | IDNA.CHECK_CONTEXTJ
    | IDNA.CHECK_CONTEXTO
    | IDNA.USE_STD3_RULES);
IDNA.Info info = new IDNA.Info();
String localpart = email.substring(0, email.lastIndexOf("@"));
String domain = email.substring(email.lastIndexOf("@") + 1);
StringBuilder output = new StringBuilder();
validator.nameToASCII(domain, output, info);
email = localpart + "@" + output.toString();

if (isEmailValid(email)) {
  subscribe();
} else {
  throw new Exception("Invalid email address, please review it and submit again");
}
```

Almacenamiento de nombres de dominio y direcciones de correo electrónico

Consideraciones de base de datos

- SQL, p. ej.., MySQL, Oracle, Microsoft SQL Server.
  - Establecer nombres de dominio en max: 255 octetos, 63 octetos por etiqueta.
    - En UTF8 nativo, longitud variable.
  - Recomendación para utilizar columnas String de longitud variable.
  - Considere/verifique el controlador/herramienta de asignación relacional de objetos (ORM) si está utilizando un.
- noSQL, p. ej.., MongoDB, CouchDB, Cassandra, HBase, Redis, Riak, Neo4J.
  - Ya longitud variable UTF8.

Nombres de dominio y direcciones de correo electrónico en la plataforma Android

android.icu.text.IDNA Library

- La misma biblioteca de la UCI, integrada en el sistema operativo Android.
  - No es necesario agregar dependencias de paquetes.
  - https://developer.android.com/reference/android/icu/text/IDNA
  - Las mismas consideraciones que se han discutido antes para la biblioteca de la UCI4J.

Conclusión

Soporte de lenguajes de programación

UASG018A

LANGUAGE	LIB NAME	COMPLIANCE (%)	Type
Javascript	Idna-Uts46	85.5	IDN
Javascript	Nodemailer	84.3	Mail
Javascript	Validator	94.2	Mail
Python3	Django_Auth	48.1	Mail
Python3	Email_Validator	86.3	Mail
Python3	Encodings_Idna	67.7	IDN
Python3	Idna	100	IDN
Python3	Smtplib	84.3	Mail
Rust	Idna	87.1	IDN
Rust	Lettre	7.8	Mail

LANGUAGE	LIB NAME	COMPLIANCE (%)	Type
C	Libcurl	84.3	Mail
C	Libidn2	95.2	IDN
C#	Mailkit	84.3	Mail
C#	Microsoft	83.9	IDN
Go	Mail	100	Mail
Go	Idna	79	IDN
Go	Smtp	19.6	Mail
Java	Commons-Validator	85.5	Mail, IDN
Java	Guava	77.8	IDN
Java	ICU	93.5	IDN
Java	Jakartamail	82.4	Mail
Java	JRE	71	IDN

Conclusión

- Tenga en cuenta que es posible que los identificadores de UA no sean totalmente compatibles con software y bibliotecas.
- Utilice las bibliotecas y marcos adecuados.
- Adapte el código para admitir correctamente UA.
- Realice pruebas de unidad y sistema utilizando casos de prueba de UA para asegurarse de que su software está listo para UA.

¡Participen!

¡Participen!

- Para obtener más información sobre UA, envíe un correo electrónico info@uasg.tech o UAProgram@icann.org
- Acceder a todos los documentos y presentaciones de la UASG: https://uasg.tech

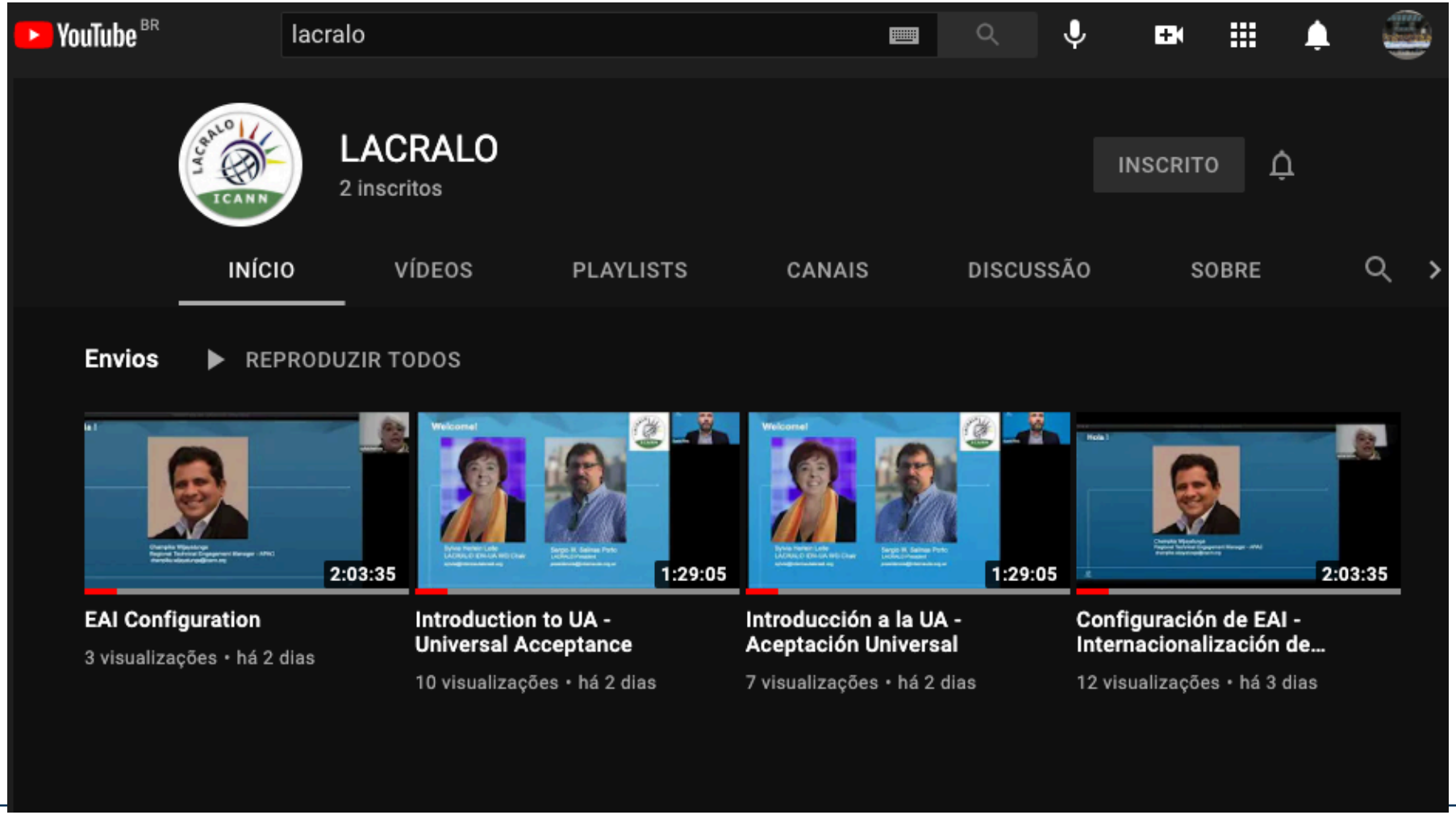
- Acceda a los detalles del trabajo en curso desde páginas wiki: https://community.icann.org/display/TUA
- Regístrese para participar o escuchar en la lista de discusión de UA en: https://uasg.tech/subscribe
- Regístrese para participar en grupos de trabajo de UA here.

Algunos materiales relevantes

- Ver https://uasg.tech para una lista completa de informes.
  - Guía rápida de aceptación universal: UASG005
  - Introducción a aceptación universal: UASG007
  - Guía rápida de EAI: UASG014
  - EAI – Una visión general técnica: UASG012
  - Cumplimiento por parte de UA de algunas bibliotecas y marcos lingüísticos de programación – UASG018A
  - EAI – Evaluación de los principales programas y servicios de correo electrónico: UASG021B
  - Marco de preparación para la aceptación universal: UASG026
  - Consideraciones para nombrar buzones de correo electrónico internacionalizados: UASG028
  - Evaluación del soporte de la EAI en el informe de software y servicios de correo electrónico: UASG030
  - UA de Sistemas de Gestión de Contenidos (CMS) Fase 1 – WordPress: UASG032

Canal Youtube de LACRALO

- https://www.youtube.com/channel/UCyNBlb2DHOq-7jbuv-i9hHg



Gracias y preguntas

One World, One Internet

Visit us at **icann.org**

Email: sarmad.hussain@icann.org

@icann

facebook.com/icannorg

youtube.com/icannnews

flickr.com/icann

linkedin/company/icann

slideshare/icannpresentations

soundcloud/icann