

**UNIVERSIDAD DE CHILE**  
**FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS**  
**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**EXTENSIÓN DE NOMBRES DE DOMINIO BAJO .CL**

**MARIO LEYTON L.**

2005

UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTENSIÓN DE NOMBRES DE DOMINIO BAJO .CL

MARIO HERNÁN GUILLERMO LEYTON LUEJE

COMISIÓN EXAMINADORA	NOTA (n°)	CALIFICACIONES: (Letras)	FIRMA
PROFESOR GUÍA SR. JOSÉ MIGUEL PIQUER	:	.....	.....
PROFESOR CO-GUÍA SR. PATRICIO POBLETE	:	.....	.....
PROFESOR INTEGRANTE SR. CÉSAR GUERRERO	:	.....	.....
NOTA FINAL EXAMEN DE TÍTULO	:	.....	.....

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL EN COMPUTACIÓN

SANTIAGO DE CHILE  
ABRIL DEL 2005

# Resumen

El presente trabajo tiene como principal objetivo extender los caracteres aceptados en la inscripción de dominios bajo .CL. Tradicionalmente el DNS ha utilizado solamente los caracteres del alfabeto inglés (a..z), el guión (-), y los dígitos (0..9). Esta limitación no permite representar en el DNS idiomas que utilizan otros caracteres. En el caso del español, por ejemplo, no es posible representar palabras con tildes, diéresis o ñ.

Actualmente existe un incipiente estándar desarrollado por la IETF, denominado IDN, que permite codificar caracteres adicionales en el DNS. El administrador del Registro .CL: NIC Chile, a cargo del Departamento de Ciencias de la Computación de la Universidad de Chile, se encuentra motivado por contar con este estándar. Esto representa un desafío, ya que la actual plataforma operacional fue ideada para trabajar solamente con los caracteres convencionales del DNS.

Para generar antecedentes se investigaron los casos de adopción de IDN en el mundo. A la fecha son pocos los registros que han adoptado este estándar. En primer lugar se identificó que existe la posibilidad de adoptar los nuevos caracteres utilizando el esquema de inscripción regular, o utilizando un período de preferencia denominado *sunrise*. Además, se diseñó un modelo de adopción de variantes, donde se definen caracteres equivalentes. En el modelo de variantes, los dominios generados a partir de caracteres equivalentes se tratan como un grupo o clase.

Los conceptos anteriores fueron plasmados en la biblioteca NIC::IDN. Esta biblioteca brinda las funcionalidades fundamentales para incorporar IDN en la plataforma operacional de NIC Chile. Además, se implementó una serie de *tests* automatizados que ayudan a verificar el correcto funcionamiento de la plataforma. La biblioteca NIC::IDN, junto con los *tests*, permitieron finalmente adaptar la plataforma para operar con caracteres extendidos.

Quisiera agradecer al equipo de NIC Chile. Por su apoyo, dedicación y profesionalismo. En especial: a Cristian Rojas por estar siempre dispuesto a debatir y cuestionar; a Hugo Salgado por su ayuda con Perl; a Eduardo Mercader por su rigurosidad; a Sebastián Castro por sus consejos y por responder estoicamente duda, tras duda[, tras duda]\*.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Justificación . . . . .	1
1.2. Objetivo General . . . . .	3
1.3. Objetivos Específicos . . . . .	3
<b>2. Antecedentes Históricos</b>	<b>4</b>
2.1. Introducción . . . . .	4
2.2. Globalización vs. Localización . . . . .	4
2.3. Experiencias de Adopción . . . . .	6
2.3.1. Inscripción Regular (sin pre-registro) . . . . .	6
2.3.1.1. Casos Polonia, Alemania y Austria . . . . .	6
2.3.2. Período de <i>Sunrise</i> . . . . .	7
2.3.2.1. Caso Corea . . . . .	8
2.4. Esquema de Variantes de Caracteres . . . . .	9
2.4.1. Caso Taiwan . . . . .	10

<b>3. Antecedentes Tecnológicos</b>	<b>11</b>
3.1. IDNA . . . . .	11
3.1.1. Definición de Conceptos . . . . .	11
3.1.2. Introducción . . . . .	12
3.1.3. ToASCII . . . . .	13
3.1.4. ToUnicode . . . . .	15
3.1.5. Limitaciones de IDNA . . . . .	16
3.1.6. IDN en Navegadores . . . . .	16
3.2. GNU LibIDN . . . . .	17
3.2.1. Descripción . . . . .	17
3.2.2. Principales Funcionalidades . . . . .	17
3.2.3. Módulo Perl, Net LibIDN . . . . .	18
3.3. Codificaciones en Perl . . . . .	18
<b>4. Diseño Adopción IDN</b>	<b>20</b>
4.1. Introducción . . . . .	20
4.2. Requerimientos . . . . .	20
4.2.1. Plataforma Operacional . . . . .	20
4.2.2. Lenguajes . . . . .	21
4.3. Escenarios para la Adopción de IDN en NIC Chile . . . . .	21

4.4. Interfaces . . . . .	22
4.4.1. Páginas Web (HTML) . . . . .	22
4.4.1.1. Despliegue . . . . .	22
4.4.1.2. Entrada . . . . .	24
4.4.2. Email . . . . .	25
4.4.2.1. Envío . . . . .	25
4.4.2.2. Recepción . . . . .	26
4.4.3. Correo (snail) . . . . .	26
4.4.4. Facturas y Cupones (PDF) . . . . .	27
4.4.5. Whois . . . . .	27
4.5. Almacenamiento, ¿IDNA, Unicode o ISO8859-1? . . . . .	28
4.5.1. Formulario de Dominio . . . . .	28
4.5.2. Sistema de Archivos ( <i>File System</i> ) . . . . .	30
4.5.3. Base de Datos (MySQL) . . . . .	31
4.5.4. Recomendación . . . . .	31
4.6. Modelo Variantes de Caracteres . . . . .	32
4.6.1. Definición Tabla de Variantes . . . . .	32
4.6.2. Dominio Canónico, Equivalencias y Clases . . . . .	33
4.6.2.1. Partición Forzada de la Clase . . . . .	35

4.6.3.	Operaciones . . . . .	36
4.6.3.1.	Proceso de Inscripción y Asignación . . . . .	36
4.6.3.2.	Renovaciones . . . . .	37
4.6.3.3.	Conflictos y Revocaciones . . . . .	37
4.6.3.4.	Proceso de Activación de Variantes . . . . .	37
4.6.3.5.	Desactivaciones . . . . .	38
4.6.3.6.	Transferencias . . . . .	38
4.6.3.7.	Eliminaciones . . . . .	38
4.6.3.8.	Modificaciones . . . . .	38
4.6.4.	Costos y Tarifas . . . . .	38
4.7.	<i>Sunrise</i> con Variantes . . . . .	40
4.7.1.	Primera Etapa . . . . .	40
4.7.1.1.	Eliminaciones, Revocaciones y Transferencias . . . . .	41
4.7.2.	Segunda Etapa (Transición) . . . . .	41
4.7.3.	Tercera Etapa (Inscripción Regular) . . . . .	42
4.8.	Modelo Biblioteca NIC::IDN . . . . .	42
4.8.1.	Descripción . . . . .	42
4.8.2.	Requerimientos Funcionalidades Básicas . . . . .	43
4.8.3.	Requerimientos Funcionalidades Extendidas (Modelo Variantes de Caracteres) . . . . .	44



<b>5. Implementación y Testing Biblioteca NIC::IDN</b>	<b>45</b>
5.1. Introducción . . . . .	45
5.2. Metodología . . . . .	45
5.3. Implementación del Módulo Perl NIC::IDN.pm . . . . .	46
5.3.1. Tablas Editables . . . . .	46
5.3.2. Hash Canónico->Variantes . . . . .	47
5.3.3. Hash Variante->Canónico . . . . .	49
5.4. Testing de la Biblioteca . . . . .	50
<b>6. Implementación y Testing, Adopción de IDN en la plataforma NIC1</b>	<b>51</b>
6.1. Introducción . . . . .	51
6.2. Módulos NIC1 . . . . .	52
6.2.1. Form . . . . .	53
6.2.2. Ingresa Solicitud . . . . .	53
6.2.3. Receive Mail . . . . .	54
6.2.4. CLIO . . . . .	54
6.2.5. Dom-CL . . . . .	54
6.2.6. Show Form . . . . .	54
6.2.7. Ultimos . . . . .	54
6.2.8. Util . . . . .	55

6.2.9.	Zone Grep y LogGrep . . . . .	55
6.3.	Operaciones . . . . .	55
6.3.1.	Formularios de Operaciones . . . . .	55
6.3.2.	Operaciones sobre Formularios . . . . .	56
6.3.2.1.	Activar . . . . .	56
6.3.2.2.	Desactivar . . . . .	57
6.3.2.3.	Asignar . . . . .	57
6.3.2.4.	Borrar y Desistir . . . . .	58
6.3.2.5.	Modificar . . . . .	58
6.3.2.6.	Pagar . . . . .	58
6.3.2.7.	Facturar . . . . .	59
6.3.2.8.	Congelar . . . . .	59
6.3.2.9.	Proteger . . . . .	60
6.4.	Misceláneos . . . . .	60
<b>7.</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>62</b>
7.1.	Conclusiones . . . . .	62
7.2.	Trabajo Futuro . . . . .	63
<b>A.</b>	<b>Guía ICANN para implementación de IDN.</b>	<b>66</b>

<b>B. API Biblioteca NIC::IDN</b>	<b>68</b>
B.1. Validar Dominio o Variante . . . . .	68
B.2. Representación Canónica . . . . .	69
B.3. Identificador de Dominio . . . . .	69
B.4. Pertencia Misma Clase . . . . .	69
B.5. Equivalencia Canónica . . . . .	70
B.6. Equivalencia . . . . .	70
B.7. Obtener Clase . . . . .	70
B.8. Pertenencia a una Clase Forzada . . . . .	71
B.9. Determinar si un dominio es IDN . . . . .	71
B.10. Determinar si se puede codificar en ACE . . . . .	71
<b>C. Listado de Archivos Modificados o Creados</b>	<b>72</b>
C.1. Host Virtual www.nic.cl . . . . .	72
C.2. Directorio CLCORREO . . . . .	73
C.3. Módulos de Perl . . . . .	73
<b>D. Documentos PDF</b>	<b>75</b>
D.1. Cupón de Pago . . . . .	75
D.2. Factura Electrónica . . . . .	76

# Capítulo 1

## Introducción

Los nombres de dominio corresponden a un identificador único en Internet. Su principal uso consiste en asociar un nombre a una dirección IP, pero no se encuentran limitados a esto. Hoy en día, los nombres de dominio se encuentran relacionados con prácticamente todos los servicios existentes en la red, y su uso continúa creciendo.

Los nombres de dominio se organizan de manera jerárquica, y además de los genéricos (TLD)<sup>1</sup>, existe uno asociado a cada país (ccTLD)<sup>2</sup>. En el caso de Chile, este corresponde a .CL, administrado actualmente por NIC Chile<sup>3</sup>, en la Universidad de Chile.

La misión de NIC Chile consiste en administrar el registro de nombres bajo .CL, rol que ha ejercido desde el año 1986, cuando IANA<sup>4</sup> delegó el ejercicio de dicha función en el Departamento de Ciencias de la Computación de la Universidad de Chile.

### 1.1. Justificación

En la actualidad los nombres de dominio inscritos en NIC Chile solamente pueden estar compuestos por los siguiente caracteres:

---

<sup>1</sup>Top Level Domain: .com, .org, .net, .biz, etc...

<sup>2</sup>Country Code Top Level Domain

<sup>3</sup><http://www.nic.cl>

<sup>4</sup>Internet Assigned Numbers Authority

abcdefghijklmnopqrstuvwxy

ABCDEFGHIJKLMNPOQRSTUVWXYZ<sup>5</sup>

0123456789

- (guión)

Con estos caracteres es posible crear un gran número de palabras del alfabeto español, pero no todas.

Las palabras que contienen el carácter “ñ”, y/o tildes: “á, é, í, ó, ú, ü”, no pueden ser representadas utilizando el DNS<sup>6</sup> convencional. Tampoco es posible utilizar caracteres simbólicos como por ejemplo: “\$, ©, +, ®”, entre otros.

NIC Chile, en su constante interés por mejorar sus servicios, se encuentra motivado en implementar esta nueva funcionalidad bajo .CL utilizando el estándar IDN<sup>7</sup>. El problema dista de ser una trivialidad, sino que más bien, presenta una serie de aristas que van desde lo técnico, a través de lo cultural y económico.

A nivel internacional, el uso de IDN aún es incipiente. Con una especificación técnica de poco más de un año, son pocos los registros TLD que han adoptado esta nueva tecnología<sup>8</sup>. Siendo menos, aquellos ccTLD que sí lo han hecho. En el caso de países de habla hispana, a la fecha todavía no existe un registro de dominios con IDN.

La inmadurez de las especificaciones, falta de compatibilidad con navegadores y aplicaciones en general además de las dificultades de integración, son algunas de las razones que explican la lenta adopción de IDN.

---

<sup>5</sup> Actualmente no se diferencian las mayúsculas de las minúsculas, pero son caracteres válidos.

<sup>6</sup> Domain Name System

<sup>7</sup> Internationalized Domain Names

<sup>8</sup> <http://www.verisign.com/nds/naming/idn/register/custorig.html>

## 1.2. Objetivo General

El objetivo de esta memoria, consiste en extender el conjunto de caracteres de nombres de dominio en NIC Chile. Esta extensión se debe aplicar en la plataforma operacional actual. Además, debe estar en acorde con las políticas definidas por NIC Chile.

## 1.3. Objetivos Específicos

- Apoyar la definición estratégica de adopción de IDN en NIC Chile.
- Soportar una definición arbitraria del alfabeto<sup>9</sup>, por parte de NIC Chile.
- Resolver los problemas de transición, desde la inscripción convencional hasta la nueva extendida, aplicando las políticas definidas por NIC Chile.
- Habilitar la inscripción y administración de IDN en la actual plataforma operacional de NIC Chile, manteniendo la compatibilidad con los dominios existentes.

---

<sup>9</sup>Subconjunto de caracteres IDN.

# Capítulo 2

## Antecedentes Históricos

### 2.1. Introducción

En este capítulo se intenta pincelar las diferentes alternativas para la adopción de IDN. Cada alternativa presenta sus propias virtudes y complejidades. Como se discutirá a continuación, no existe un enfoque erróneo o correcto. Mas bien, cada Registro<sup>1</sup> debe considerar la características culturales y lingüísticas de su localidad al momento de adoptar una solución.

### 2.2. Globalización vs. Localización

No cabe duda que la globalización es poderosa. Hoy en día es posible enviar un mensaje desde un equipo móvil a cualquier parte del mundo. Esto es maravilloso. ¿ O no ? La verdad es que no, pues la mayor parte del planeta no utiliza el alfabeto inglés. Por esta razón es importante recordar que en una sociedad cada vez más globalizada, resulta cada vez más difícil mantener una identidad cultural.

---

<sup>1</sup>Registro, corresponde al administrador del TLD. En el caso de Chile es NIC Chile.

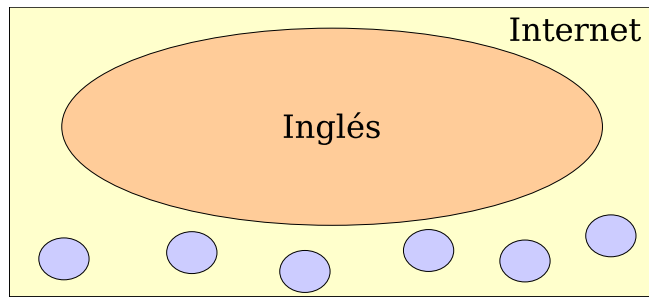


Figura 2.1: Inglés Dominante

Por otro lado, si suponemos que cada lengua resolverá sus propias dificultades locales, corremos el riesgo de lograr varias redes segmentadas en vez de una inter-red.

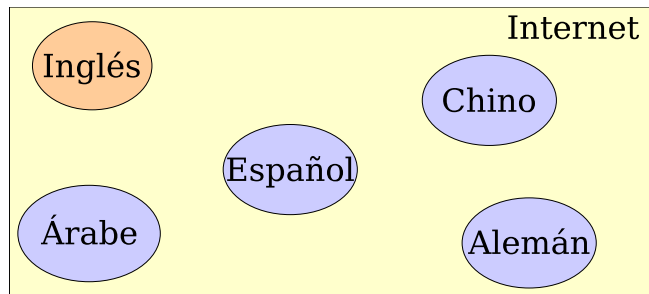


Figura 2.2: Idiomas Segmentados

En una verdadera sociedad global, un usuario debe poder comunicarse con los símbolos de su propia lengua, independiente del idioma del receptor.

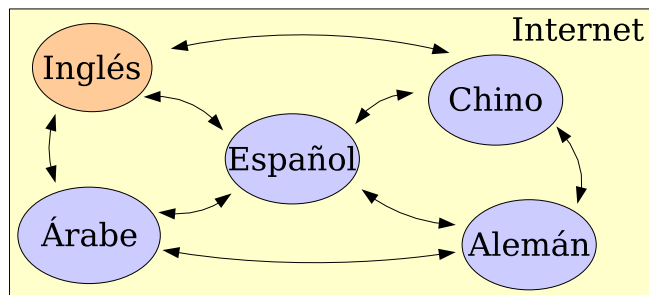


Figura 2.3: Idiomas Interoperables

En la actualidad, un usuario global de Internet no puede tener una experiencia completamente local. IDN representa un esfuerzo por dar un paso en esta dirección.



## 2.3. Experiencias de Adopción

### 2.3.1. Inscripción Regular (sin pre-registro)

Esta estrategia consiste en habilitar la inscripción de IDN para algún conjunto de caracteres extendidos. A partir de una fecha determinada, las solicitudes de registro por dominios se atienden utilizando el régimen regular de inscripción. Por ejemplo, en orden de llegada (FCFS)<sup>2</sup>.

- La principal ventaja es su simplicidad de implementación, además puede significar un aumento en el número de inscripciones dominios.
- Por otro lado, es esperable que el número de conflictos y revocaciones de dominios aumente considerablemente<sup>3</sup>. Además, se presta para el uso malintencionado de dominios similares y el *cybersquatting*. Este último corresponde a un “término utilizado para describir la práctica de registro y reclamación de derechos sobre un nombre de dominio, cuando claramente no pertenecen al solicitante. El *cybersquatter* ofrece entonces el dominio al dueño legítimo en un precio inflado, un acto que puede ser considerado extorsión”.<sup>4</sup>.

#### 2.3.1.1. Casos Polonia, Alemania y Austria

Ejemplos de este enfoque son actualmente Polonia (.PL), Alemania (.DE) y Austria (.AT). Polonia se ha enfocado en soportar la mayor cantidad de minorías locales posibles. Por esta razón, actualmente soportan los siguientes conjuntos de caracteres: *Latin-1*, *Griego*, *Hebreo*, *Árabe* y *Cirílico* [10]. Los principales aspectos de su implementación se encuentran descritos a continuación[9]:

- Registro en orden de llegada (FCFS), sin pre-registros (*sunrise*).
- No se aceptan dominios con caracteres de más de un conjunto. En otras palabras, no se puede mezclar caracteres de distintos conjuntos.
- Solamente aceptan registros con dominios codificados en ASCII, según los RFC 3490, 3491, 3492[13, 14, 15].

---

<sup>2</sup>First come First Served

<sup>3</sup>Un conflicto de dominio se genera cuando existen mas de dos solicitantes por un dominio.

<sup>4</sup>Wikipedia <http://en.wikipedia.org/wiki/Cybersquatting>

- Los dominios registrados no son etiquetados con un idioma (contrario a la Guía de ICANN).

La reglamentación de Alemania no dista significativamente de los puntos anteriores. Lo más importante en ambos casos es la ausencia de un período *sunrise* y/o *variantes de caracteres*. En el caso de Alemania, la justificación se da por dos razones[1]:

- Experiencias de .info y .biz con *sunrise* han mostrado dificultades y caos organizacional, no proporcional a los abusos de registros mal intencionados.
- Dificultades por verificar derechos sobre nombres de postulantes.

Al igual que Alemania y Polonia, Austria también opera en modo FCFS. Sus experiencias han mostrado que durante los primeros 90 minutos, después de lanzar IDN, se recibieron 78.650 solicitudes de registro. Además, los dominios más pedidos:

- bücher.at (libros),
- diät.at (dieta),
- flüge.at (vuelos)

superaron las 800 solicitudes[11].

### 2.3.2. Período de *Sunrise*

El período de *sunrise* privilegia a ciertos actores, durante algún intervalo de tiempo, con la inscripción de IDN. Posibles actores pueden ser: marcas registradas, empresas o clientes actuales. Además, el período de *sunrise* puede contar con múltiples etapas.

- Presenta la ventaja de privilegiar a los clientes actuales con una primera opción de registro, disminuyendo el número de posibles conflictos.
- Por otro lado, requiere implementar cada etapa en la plataforma operacional, aumentando la complejidad organizacional. Además, se encuentra asociado a malas experiencias en el pasado, generando abusos y *cybersquatting*.

### 2.3.2.1. Caso Corea

La implementación de IDN en Corea utilizó un período de *sunrise* con cuatro etapas[6].

- Palabras Reservadas: Las palabras reservadas pueden ser inscritas durante el período de un año por agencias gubernamentales, legislativas, educacionales, entre otras. Actualmente, a pocos meses de cumplir un año, solamente el 16 % (4.747 de 29.720) de las palabras reservadas han sido inscritas, por lo que KRNIC ha considerado ampliar el plazo.

Paralelamente a la estrategia anterior, se ejecutaron secuencialmente las siguientes etapas:

- Primera Etapa (6 Semanas): Durante esta etapa solamente se puede registrar dominios por marcas registradas y compañías, que reflejen similitud de nombre. Cada marca registrada fue autorizada para registrar un solo dominio, mientras que una empresa dos dominios. En estas seis semanas se registraron 11.650 dominios.
- Segunda Etapa (2 Semanas): Ciudades y empresas con una número de registro pueden solicitar un nombre de dominio. En esta fase se registraron 19.276 dominios.
- Tercera Etapa: A continuación, los registros se atienden en orden de llegada (FCFS).

A Julio del 2004 el número total de dominios registrados en .KR es de 613.000:

Cuadro 2.1: IDN en Corea (Julio 2004)

	Nº Dominios	Porcentaje
IDN	79.581	100 %
IDN-activos	27.645	34 %

Es decir, que después de casi un año de aceptar registros con IDN, solamente un 4.5 % del total de dominios registrados en Corea se encuentran operativos con IDN.

## 2.4. Esquema de Variantes de Caracteres

Para cada caracter se define una tabla de variantes donde se identifican los caracteres equivalentes.

Cuadro 2.2: Ejemplo Tabla Variantes[6]

VCP	TWPV	CNPV	CV
台	台	台	台權靈臺颱
網	網	网	網网
中	中	中	中
心	心	心	心

Al registrar un dominio se reservan también todos los dominios generados a partir de la tabla de variantes. Cabe destacar que en la actualidad, al momento de registrar un dominio, se obtiene exactamente un dominio. El enfoque de variante de caracteres rompe con este paradigma, ya que al momento de registrar un dominio se obtienen (o reservan), todas las combinaciones generadas a partir de la tabla de variantes.

- La principal ventaja es disminuir la posibilidad de *cybersquatting*. Además se protege a los clientes actuales, disminuyendo las expectativas de incrementos significativo en los conflictos.
- Como desventaja, no se aprovechan completamente los nombres de dominio extendidos. Por ejemplo, el titular de `papa.cl` es el único posible titular de `papá.cl`, a pesar que las palabras tienen significados distintos.

El esquema de variantes también genera interrogantes importantes para el registro de dominios:

- ¿Cuál es el costo de reservar los dominios con variantes?
- ¿Debe este costo estar incluido en el registro del dominio?
- ¿Cuál debe ser el costo de activación de dominios con variantes?

## 2.4.1. Caso Taiwan

Las variantes de caracteres no es un concepto ajeno al DNS. Por ejemplo, el DNS no distingue entre mayúsculas y minúsculas. Es decir, una variante del carácter "A" corresponde al carácter "a", y viceversa. La creación de tablas de variantes de caracteres intenta resolver problemas similares en otros alfabetos, poniendo énfasis en la localidad de cada registro.

En la actualidad, los países que han optado por el esquema de variantes de caracteres son principalmente aquellos que utilizan lenguas no romances. Entre ellos se encuentran China, Japón, Corea, Taiwan, Arabia Saudita y Túnez. La principal dificultad para estos países consiste en definir la tabla de variantes, ya que debe ser específica a su localidad<sup>5</sup>, puede tratar con varios miles de caracteres, y en ciertos casos existe más de un alfabeto para una misma lengua.

```
Original form : 台網中心.tw ( punycode : xn--fiq43lrriz83a.tw )
Traditional form : 台網中心.tw ( punycode : xn--fiq43lrriz83a.tw )
Simplified form : 台网中心.tw ( punycode: xn--fiq43lrrfy5a.tw )
Character Variant DN : 權網中心.tw 權网中心.tw 臺網中心.tw
                     臺网中心.tw 臺網中心.tw 臺网中心.tw
                     廳網中心.tw 廳网中心.tw
```

Figura 2.4: Ejemplo Caso Taiwan[18]

El caso de Taiwan se utilizan los caracteres chinos, tanto en su versión tradicional como simplificada. Al registrar un dominio el cliente obtiene el dominio solicitado, el dominio con caracteres tradicionales y el dominio con caracteres simplificados. El resto de las variantes son reservadas. El siguiente diagrama muestra las operaciones de .TW:

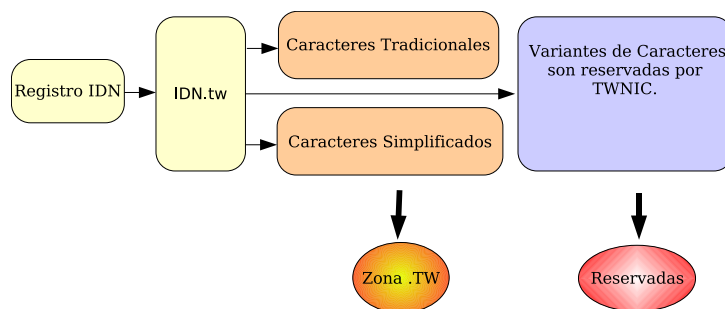


Figura 2.5: Ejemplo Proceso Inscripción Caso Taiwan

<sup>5</sup>La tabla China y de Taiwan son diferentes a pesar de compartir la mayoría de sus caracteres.

# Capítulo 3

## Antecedentes Tecnológicos

### 3.1. IDNA

#### 3.1.1. Definición de Conceptos

A continuación se definen una serie de conceptos estrechamente relacionados con la extensión de nombres de dominio.

Caracter, Representación atómica de una letra, número o símbolo.

ASCII, American Standard Code for Information Interchange, es el estándar de facto para la representación de caracteres. Originalmente utilizó 7 bits por caracter, actualmente existen extensiones de 8 bits por caracter.

Unicode, Estándar de representación de caracteres que utiliza 16 bits por caracter, generando mas de 65,000 caracteres diferentes.

Stringprep, Método de formato para caracteres, que intenta aumentar la probabilidad de éxito al ingresar y comparar dos textos con caracteres Unicode. Por ejemplo, si comparamos lexicográficamente: *Papa* y *papa*, el resultado será que corresponden a textos diferentes. Una de las labores realizadas por Stringprep consiste en convertir todas las letras a minúsculas. De esta manera, al comparar: *Stringprep(Papa)* y *Stringprep(papa)* el resultado indica que corresponden al mismo texto.

- Nameprep, Es un perfil de Stringprep, orientado al formato de IDN. Entre otros realiza: conversión del texto a minúsculas, normalización: unificación de caracteres con múltiples representaciones (Ej: Alfabeto Japonés Katakana tamaño mediano y Katakana tamaño completo), eliminación de caracteres prohibidos (Ej: espacio), eliminación de ambigüedad entre caracteres en textos bidireccionales (Ej: Hebreo, Árabe) y verificación de caracteres Unicode asignados.
- Bootstring, Algoritmo que permite representar un conjunto de caracteres a través de un conjunto de menor tamaño. Por ejemplo, si se desea representar las letras {a,b,c,d} solamente con {a,b}, es posible definir que cada letra utiliza dos caracteres y que {a<->aa, b<->bb, c<->ab, d<->ba}.
- Punycode, Instancia de Bootstring especialmente orientada hacia la codificación de IDN en caracteres ASCII. Entre sus virtudes se destacan<sup>1</sup>: independencia de idioma, superioridad de compresión, código compacto y superioridad en codificación de caracteres Japoneses, Chinos y Coreanos.
- ACE, “ASCII Compatible Encoding”, corresponde a la codificación ASCII compatible de un dominio después de aplicar el algoritmo ToASCII. Los dominios ACE son identificados mediante un prefijo que tiene la forma “xn-”.

### 3.1.2. Introducción

IDNA<sup>2</sup> es un metodología estándar para manejo de IDN en aplicaciones. Como principal característica, IDNA opera con compatibilidad hacia atrás mediante la integración de los conceptos definidos en la sección anterior. Utilizando IDNA, es posible integrar IDN en aplicaciones desarrolladas para operar solamente con caracteres ASCII. El principal ejemplo de su aplicación es el Sistema de Nombres de Dominio, DNS. Donde, por ejemplo, `viña.cl` es transformado en `xn--via-8ma.cl` al ser insertado en el DNS.

---

<sup>1</sup>Punycode fue la elección de la IETF para IDNA por sobre otros algoritmos propuestos, como por ejemplo RACE de Verisign.

<sup>2</sup>del inglés, Internacionalización de Nombres de Dominio en Aplicaciones.

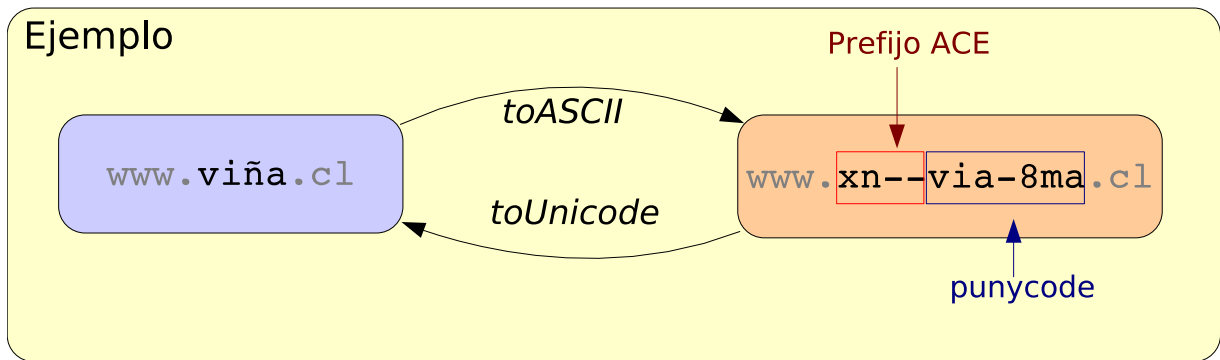


Figura 3.1: Esquema IDNA

Utilizando ToASCII y ToUnicode, la aplicación no consciente de IDN puede seguir operando. Los dominios son convertidos en ASCII antes de ingresar a la aplicación, y deben ser devueltos a Unicode siempre que se desplieguen al usuario.

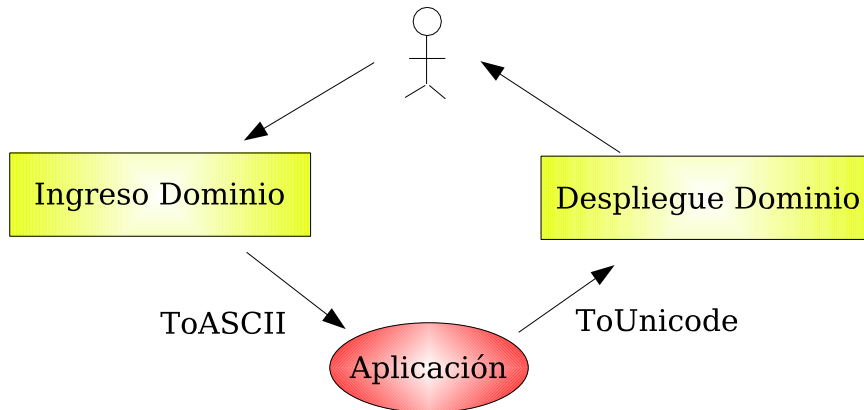


Figura 3.2: Esquema IDNA

### 3.1.3. ToASCII

ToASCII es un algoritmo que intenta convertir un nombre de dominio a caracteres ASCII. Utiliza Nameprep y Punycode. Puede tener tres posibles resultados: el nombre de dominio no es convertido ya que no lo requiere, el nombre de dominio es convertido, o un error. Este último caso ocurre, por ejemplo, cuando el largo total de la nueva representación excede 63 caracteres.

A continuación se presenta un diagrama del algoritmo ToASCII generado a partir del RFC-3490[13]:



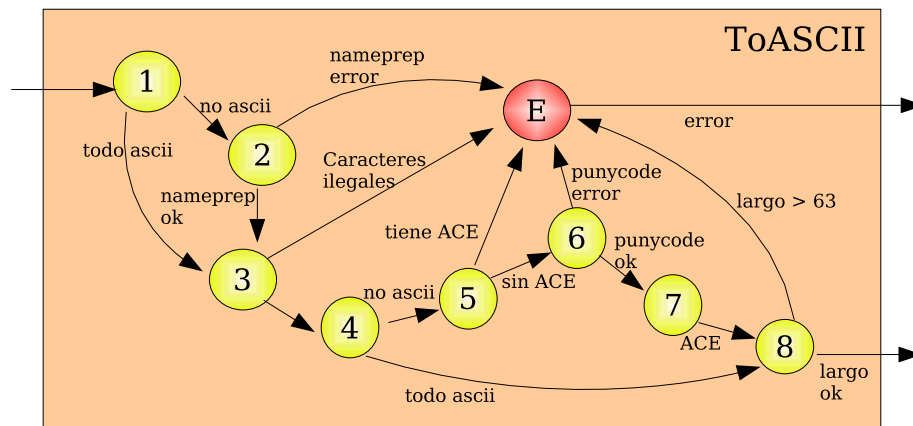


Figura 3.3: Algoritmo ToASCII

Como se aprecia, el algoritmo puede retornar un error, o el texto transformado en ASCII. El algoritmo debe ser aplicado para cada bloque (*label*) de un dominio por separado. Es decir, que para *www.ejemplo.cl*, se debe aplicar primero a *www*, luego a *ejemplo*, y finalmente a *cl*.

A continuación se presenta el proceso de conversión para el texto Papá.

1. Papá contiene caracteres no ASCII, por lo que debe pasar a 2.
2. Papá es normalizado exitosamente utilizando Nameprep: papá.
3. papá no contiene caracteres ilegales como el punto (.), entre otros.
4. papá contiene caracteres no ASCII, debe pasar a 5.
5. papá no comienza con prefijo ACE.
6. papá es codificado exitosamente en ASCII mediante Punycode: pap-gla
7. pap-gla es marcado con prefijo ACE: xn--pap-gla
8. xn--pap-gla no supera 63 caracteres de largo, se retorna el texto transformado.

Por último, se puede observar que el algoritmo cuenta con algunas propiedades útiles:

- `toASCII(texto)==toASCII(toASCII(texto))`
- `textoASCII==toASCII(textoASCII)`, siempre y cuando `textoascii` sea válido.
- `textoACE==toASCII(textoUnicode)`

### 3.1.4. ToUnicode

ToUnicode es un algoritmo que convierte un nombre de dominio en su equivalente Unicode. Este algoritmo nunca falla. En caso de error retorna el dominio original. Específicamente, permite convertir nombres de dominio previamente transformados con el algoritmo ToASCII.

A continuación se presenta una figura con el diagrama del algoritmo ToUnicode generada a partir del RFC-3490[13]:

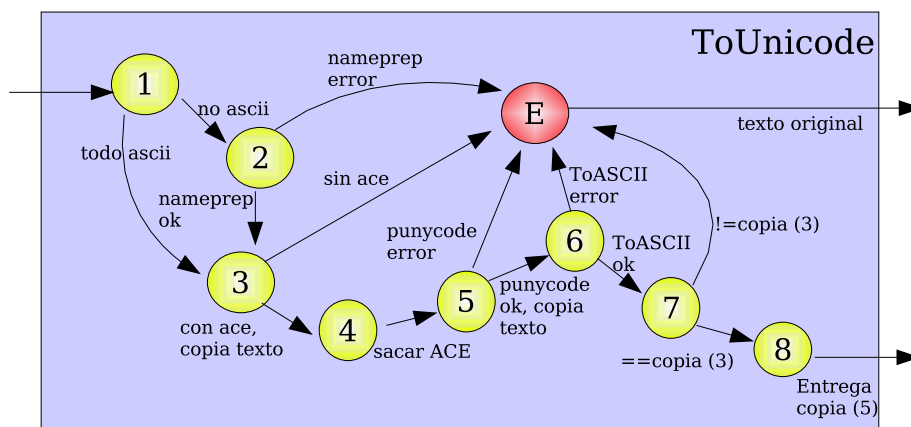


Figura 3.4: Algoritmo ToUnicode

De igual manera que ToASCII, el algoritmo ToUnicode debe ser aplicado a cada bloque de un dominio. El proceso de transformación de *xn--pap-gla* (*Papá* previamente codificado mediante ToASCII) en Unicode se presenta a continuación:

1. *xn--pap-gla* contiene solamente caracteres ASCII, se debe pasar 3.
2. (Paso no aplicable en este ejemplo)
3. *xn--pap-gla* verificación existencia prefijo ACE.
4. *xn--pap-gla* el prefijo ACE es extraído: *pap-gla*
5. *pap-gla* es transformado mediante Punycode: *papá*
6. *papá* es transformado con ToASCII: *xn--pap-gla*
7. El resultado de 6 es comparado con la entrada de 3.

8. Si la comparación es exitosa, se retorna el resultado de 5. De lo contrario se retorna el texto original.

Las propiedades que cumple este algoritmo son:

- `toUnicode(texto)==toUnicode(toUnicode(texto))`
- `textoUnicode==toUnicode(textoACE)`
- `textoASCII==toUnicode(textoASCII)`

### 3.1.5. Limitaciones de IDNA

La principal limitación de IDNA, es que no resuelve problemas lingüísticos. Por ejemplo, palabras que se diferencian por un solo caracter extendido: papá, papa. Tampoco resuelve el problema de palabras con múltiples significados: Papa (Autoridad Eclesiástica) y papa (tubérculo). Este tipo de limitaciones deberán ser enfrentadas por las definiciones políticas de NIC Chile.

### 3.1.6. IDN en Navegadores

Una vez que un dominio IDN es obtenido a partir del DNS, se debe convertir en Unicode (mediante el algoritmo ToUnicode), para ser desplegado al usuario final. A partir de las versiones especificadas, los siguientes navegadores soportan IDN: Netscape 7, Mozilla 1.4, Opera 7. Microsoft Explorer no soportan IDN de manera nativa. Sin embargo, Verisign<sup>3</sup> ha desarrollado un plug-in que agrega esta funcionalidad: *i-Nav*<sup>4</sup>. El plug-in es gratuito y funciona a partir de Explorer 5.0.

---

<sup>3</sup>Verisign administra actualmente .com y .net

<sup>4</sup><http://www.idnnow.com/>

## 3.2. GNU LibIDN

### 3.2.1. Descripción

La biblioteca LibIDN[7] escrita principalmente por Simon Joseffson, corresponde a un esfuerzo de software libre por implementar el estándar IDN[13, 14, 15]. Actualmente se encuentra respaldada por el “*GNU Project*”<sup>5</sup> y está compuesta por dos implementaciones, una en *C* y otra en *Java*. Las principales razones para elegir esta implementación por sobre sus pares es:

- Licencia de acuerdo con las necesidades y enfoque de NIC Chile: *LGPL*<sup>6</sup> (software libre que puede ser vinculado desde código propietario).
- Respaldo del proyecto GNU.
- Implementación en más de un lenguaje: *C* y *Java*.
- Existencia de módulo *Perl* para encapsular la biblioteca :“`Net::LibIDN`”.
- Desarrollo activo y actualizaciones frecuentes.

### 3.2.2. Principales Funcionalidades

Esta biblioteca presenta una serie de funcionalidades relacionadas con el estándar IDN. Las principales funcionalidades corresponden a la implementación de los algoritmos ToASCII y ToUnicode:

```
/****** JAVA *****/
package gnu.inet.encoding;
public static String toASCII(String input);
public static String toUnicode(String input);

/****** C *****/
int idna_to_ascii_8z (const char *input, char **output, int flags);
idna_to_unicode_8z1z (const char *input, char **output, int flags);
```

---

<sup>5</sup><http://www.gnu.org>

<sup>6</sup>Lesser General Public License

### 3.2.3. Módulo Perl, Net LibIDN

El módulo de Perl opera como *wrapper* para la implementación de C de la biblioteca GNU LibIDN. El API es el siguiente:

```
Net::LibIDN::idn_to_ascii(VAR1, VAR2)
Net::LibIDN::idn_to_unicode(VAR1, VAR2)

#VAR1: String de caracteres.
#VAR2: "UTF-8" o "ISO8859-1" (Representacion de VAR1).
```

Donde la primera variable corresponde al texto que se desea transformar, y la segunda indica la representación actual de la primera variable.

## 3.3. Codificaciones en Perl

El estado de las codificaciones en Perl se encuentra relativamente confuso debido a la adopción de Unicode. En esta sección se intenta esclarecer el modelo de codificación utilizado en Perl, ya que es de vital importancia comprenderlo para poder operar con caracteres extendidos.

Actualmente en Perl, al imprimir un texto se obtienen por defecto caracteres en las siguientes codificaciones:

Cuadro 3.1: Codificaciones en Perl

Rango	Codificación
0-127	ASCII (UTF-8 compatible)
128-255	ISO8859-1
256-...	UTF-8

Cuando un texto contiene solamente caracteres menores que 256, entonces el texto es impreso en ISO8859-1. Si el texto contiene algún carácter mayor, todos los caracteres serán impresos en UTF-8. El objetivo de este modelo es mantener la compatibilidad hacia atrás con ISO8859-1 incorporando a la vez los caracteres Unicode.

Afortunadamente, a partir de la versión 5.8 de Perl, si se desea imprimir un texto en una sola codificación se puede hacer. Para esto se debe especificar una codificación en el *filehandle*.

```
open FILE, ">:utf8", $filename;
```

Análogamente, al momento de leer un archivo se debe ayudar a Perl indicando la codificación de dicho archivo.

```
open FILE, "<:encoding(iso-8859-7)", $filename;
```

Al leer del archivo, Perl se encargará de transformar el texto leído en la representación interna de Perl.

Cabe notar que al manipular *strings* en Perl, el largo de un *string* puede no ser el mismo que el tamaño en bytes. Esto se debe a que caracteres codificados con UTF-8 pueden utilizar más de 1 byte. Sin embargo esto no es un problema cuando Perl tiene correctamente identificado la codificación del *string*, ya que las funciones de manejo funcionarán adecuadamente dependiendo de la codificación. Por ejemplo, la función `length()`, retorna el número de caracteres y no el tamaño en bytes.

Para las ocasiones en que Perl no tenga identificado correctamente la codificación del algún *string*, es posible manipular la interpretación de Perl utilizando el módulo `Encode`.

```
require Encode;
```

```
my $string = Encode::decode_utf8( $string );
```

Este ejemplo indica a Perl que interprete la variable como utf8.

# Capítulo 4

## Diseño Adopción IDN

### 4.1. Introducción

En este capítulo se revisan las principales características que debe tener el modelo de adopción de IDN en NIC Chile. Este modelo intenta abarcar todos los aspectos necesarios para incorporar IDN en la plataforma operacional, y es esperable que sirva como referencia para el trabajo futuro (sección 7.2).

### 4.2. Requerimientos

#### 4.2.1. Plataforma Operacional

En la actualidad NIC Chile cuenta con una plataforma operacional que supera las 100.000 líneas de código, desarrollada a lo largo de 12 años, que llamaremos NIC1. Dicha plataforma continúa creciendo para satisfacer las nuevas necesidades que surgen en la administración de nombres de dominio. Paralelamente, se encuentra en desarrollo una nueva plataforma operacional denominada NIC3.

## 4.2.2. Lenguajes

En la plataforma operacional NIC1, el lenguaje utilizado principalmente (con algunas excepciones) corresponde a *Perl*. Esto debe ser tomado en consideración tanto en el diseño del modelo como en la implementación.

En NIC3, el lenguaje de implementación corresponde a *Java* y *Perl*, por lo que se debe diseñar tomando en consideración que en el futuro deberá existir también una implementación en *Java*.

## 4.3. Escenarios para la Adopción de IDN en NIC Chile

En el Capítulo de Antecedentes Históricos (Sección 2) se discutieron las experiencias de diferentes ccTLD al adoptar IDN. De estas experiencias se desprende que existen dos posibles enfoques de adopción:

- Inscripción Regular
- Inscripción con *Sunrise* (Pre-registro)

Además, en ambos casos es posible utilizar el Modelo de Variantes.

El trabajo necesario para implementar los diferentes enfoques se puede descomponer en:

1. Adaptar la plataforma NIC1 para el uso de IDN. Es decir, compatibilizar las interfaces (Sección 4.4) y el almacenamiento (Sección 4.5) con caracteres extendidos.
2. Aplicar el Modelo de Variantes (Sección 4.6).
3. Planificar etapas del *sunrise*. (Sección 4.7).

Por lo tanto, las cuatro posibles alternativas de adopción están descritas por:



Cuadro 4.1: Alternativas de Adopción

	Sin Variantes	Con Variantes
Inscripción Regular	(1)	(1,2)
Inscripción con <i>Sunrise</i>	(1,3)	(1,2,3)

De los escenarios descartaremos (1,3) ya que posiblemente requiere utilizar algún esquema de validación no automatizado. Por la misma razón, en este documento se asumirá que el escenario (1,2,3) corresponde al uso de variantes solamente durante el *sunrise*, dejando de ser utilizadas una vez que termine este período de pre-registro.

## 4.4. Interfaces

Las interfaces no sólo corresponden al despliegue de nombres de dominio ante el usuario, sino que también están relacionadas con el ingreso de los datos por parte del usuario.

Debido a las diferentes codificaciones de caracteres en boga, es importante que el usuario efectivamente esté mirando lo que queremos desplegarle. Y que al momento de ingresar un dato, se reciba lo que el usuario cree haber ingresado.

Por esta razón a lo largo de esta sección, se discutirán las diferentes interfaces con los usuarios y los problemas que surgen al utilizar caracteres fuera del ASCII tradicional.

### 4.4.1. Páginas Web (HTML)

#### 4.4.1.1. Despliegue

La codificación del contenido de las página web es negociada por el protocolo HTTP. En general esta codificación corresponde a ISO8859-1, pero se puede especificar otra como UTF-8. El siguiente ejemplo muestra los encabezados de una conexión HTTP:

```
telnet www.nic.cl 80
```

```
Trying 200.1.123.3...
Connected to www.nic.cl.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.nic.cl

HTTP/1.1 200 OK
Date: Thu, 16 Dec 2004 20:05:11 GMT
Server: Apache
Last-Modified: Tue, 14 Dec 2004 13:28:36 GMT
Accept-Ranges: bytes
Content-Length: 14302
Content-Type: text/html; charset=ISO-8859-1
```

Como se aprecia en la última línea, actualmente la codificación de NIC Chile se especifica como: `charset=ISO-8859-1`. Alternativamente la codificación puede ser configurada en el encabezado HTML agregando una *meta-tag* con el charset, como es el caso Google Taiwan ([www.google.com.tw](http://www.google.com.tw)):

```
<meta http-equiv="content-type" content="text/html; charset=Big5">
```

Es importante recordar que finalmente dependerá del navegador interpretar correctamente la codificación. Por lo tanto, la compatibilidad de navegadores con la codificación utilizada es importante. En el caso de NIC Chile las páginas pueden estar en alguna de las siguientes codificaciones:

- Despliegue en ASCII con caracteres escapados. Por ejemplo: `pedigüeñería.cl` se convierte en `pedig&#252e&#241er&#237a.cl`. Con la ventaja de alta compatibilidad, a costa de transformar los caracteres cada vez que se desean imprimir.
- Despliegue en ISO8859-1. La ventaja es que los caracteres se escriben directamente, sin la necesidad de ser escapados. Pero la desventaja es que solamente se puede representar un pequeño subconjunto de todos los caracteres Unicode.
- Despliegue en Unicode (UTF-8). Las páginas se sirven directamente en UTF-8. No es necesario realizar un proceso de conversión si es que el contenido ya se encuentra en UTF-8. Puede presentar problemas con algunos navegadores muy antiguos.

Una cuarta posibilidad consisten en mostrar los dominios como imágenes. Esto puede resultar especialmente útil al momento de registrar dominios, para que durante el proceso de inscripción, el usuario pueda confirmar el dominio que solicita.

La recomendación de este documento es utilizar ISO8859-1, mientras los caracteres extendidos lo permitan. Eventualmente será necesario realizar el proceso de conversión a UTF-8 cuando se estime conveniente.

#### 4.4.1.2. Entrada

Cuando un usuario ingresa datos en un formulario HTML, ya sea por el método GET o POST el navegador especifica un `Content-Type` asociado. Como por ejemplo:

```
Content-Type: application/x-www-form-urlencoded
```

Al recibir la información, el servidor web junto con el módulo de CGI se encarga de transformar dicha codificación en el *encoding* propio del lenguaje, en este caso Perl.

Un usuario también puede ingresar caracteres en la barra de direcciones URL. En un navegador consciente de IDN, si un usuario ingresa un dominio con caracteres extendidos, éste es convertido con ToASCII antes de ser resuelto mediante el DNS. En un navegador no consciente de IDN, el resultado es incierto. Un navegador puede indicar la URL como inválida. Pero también existen casos, como Explorer de Microsoft, donde el navegador realiza la consulta DNS directamente con caracteres fuera del estándar ASCII. Esto es posible, ya que cada carácter es codificado en el DNS utilizando un octeto (8 bits). El estándar ASCII solamente requiere 7 bits para ser codificado, y por lo tanto ocupa los rangos [0-127]. Es decir, que el rango de caracteres [128-255] puede ser codificado en el DNS sin que esto genere un error en el proceso de resolución de nombres.

El problema anterior es aprovechado por Verisign, actual administrador de los TLDs .com y .net, para ofrecer un servicio de redirección: *sitfinder*. Cuando el dominio.com contiene caracteres superiores al 127, los servidores autoritarios de .com redirigen la consulta a su servicio *sitfinder*, donde se ofrece la instalación del plugin *i-Nav*. Este plugin agrega compatibilidad con el estándar IDN a Microsoft Explorer.

Esta ingeniosa solución requiere incorporar esta funcionalidad lógica en los servidores de DNS, y por lo tanto significa tener bajo control todos los servidores autoritarios para la zona en cuestión. Para los dominios de tercer nivel y superiores, el actual servicio de *sitfinder* no se encuentra habilitado. Sin embargo, de acuerdo al algoritmo de resolución propuesto en la sección 4.3.2 del RFC 1034, el servicio *sitfinder* para dominio de tercer nivel y superiores podría funcionar, ya que la consulta (QNAME) incluye el nombre de dominio completo.

## 4.4.2. Email

### 4.4.2.1. Envío

Los correos automatizados que son enviados a los clientes también deberán soportar contenido con caracteres extendidos. La recomendación en este caso, es utilizar MIME para resolver este problema. Mediante MIME, los correos pueden indicar que tipo de codificación se está utilizando, para que los lectores pueden desplegar adecuadamente dichos correos. Por ejemplo, si deseamos enviar correos en ISO8859-1, se deben definir los siguientes parámetros en el encabezado de los correos:

```
MIME-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1;
```

Sin embargo, esto no resuelve todos los problemas, ya que ahora en el encabezado también pueden aparecer caracteres fuera del ASCII. Por ejemplo un encabezado de la forma:

```
To: test@viña.cl.
```

La solución puede ser convertir el dominio en su equivalente ACE al momento de enviar el correo, y escribir el encabezado como: `To: test@xn--via-8ma.cl`. Sin embargo, se requieren clientes de correo capaces de interpretar el estándar IDN para no desplegar el ACE directamente al usuario.

Adicionalmente, es imaginable también tener correos de la forma: `viña@viña.cl`, donde tanto el dominio como la casilla (lo que está a la izquierda del símbolo “@”) contienen caracteres fuera del ASCII tradicional. Aún no existe una solución para este problema, y su discusión escapa

del campo de los nombres de dominio.

NIC Chile se ve afectado por la existencia de casillas de correo con IDN, ya que en el formulario de registro existen campos donde el usuario debe ingresar direcciones de correo como contacto. Por esta razón, se deberá decidir si se aceptan casillas de correos con dominios IDN en el formulario. En caso de aceptarse, se debe convertir manualmente las casillas de correo a ACE para poder enviarlas al usuario, ya que actualmente *sendmail*<sup>1</sup> no soporta IDN.

#### 4.4.2.2. Recepción

Además de los correos enviados se deben considerar los correos recibidos por NIC Chile. Podemos clasificar los correos en 3 tipos:

- Dirigidos a una persona del staff.
- Dirigidos a un rol de NIC Chile, (*soporte@nic.cl*, *legal@nic.cl*, *info@nic.cl*, etc...)
- Dirigidos directamente al sistema (Ej: para realizar inscripciones, modificaciones).

Este último ítem presenta complicaciones, ya que aún se encuentra en uso el proceso de registro de nombres de dominio vía email utilizando el módulo `ReceiveMail`. En caso de continuar operando con este medio de registro, se debe analizar si soporta caracteres fuera del ASCII tradicional, o alternativamente si es que se aceptarán solicitudes por dominios IDN en ACE.

#### 4.4.3. Correo (snail)

Actualmente se envía a los clientes, por correo tradicional, el cupón de renovación. Para esto se genera un listado de dominios en renovación, el cual es enviado mensualmente a Sim Courier, organización externa que imprime y envía los cupones a los asignatarios de los dominios.

Este listado siempre ha contenido caracteres fuera del ASCII tradicional, por lo que la implementación de IDN no presenta un problema mientras los caracteres puedan ser representados en

---

<sup>1</sup>Sendmail, corresponde a uno de los servidores de correos más populares en el mundo Linux/Unix.

ISO8859-1. En caso de trabajar con UTF-8 deberá notificarse a Sim Courier para que interprete la codificación del listado correctamente.

#### **4.4.4. Facturas y Cupones (PDF)**

Actualmente NIC Chile opera con el estándar de facturación electrónica. Las facturas y cupones son impresas en NIC Chile o se pueden bajar de la página web como un archivo en formato PDF. El estándar de facturación permite caracteres definidos en ISO8859-1, por lo que no todo el estándar Unicode es representable.

Si en algún momento se desea utilizar caracteres fuera del ISO8859-1, entonces una posible solución es facturar los dominios utilizando su codificación ACE.

Es esperable que eventualmente el estándar de facturación migre a UTF-8.

#### **4.4.5. Whois**

El protocolo *whois* se encuentra descrito en el RFC3912[16]. Su principal objetivo es brindar información con respecto a los servicios ofrecidos en Internet. El principal problema de *whois*, es que no presenta ningún mecanismo de internacionalización. Tradicionalmente siempre ha operado en ASCII. Hoy en día, los *whois* de diferentes organizaciones operan con distintas codificaciones de caracteres, dependiendo de sus necesidades.

Es importante notar que el *whois* no contiene *tagging* (como lo hace el email con MIME), y por lo tanto no se puede saber cual es el conjunto de caracteres utilizados por el cliente o servidor. Por esta razón es necesario decidir la codificación del servidor, pensando en la configuración de la mayoría de sus clientes.

Cuadro 4.2: Alternativas Whois

	xn-punycode	iso-8859-1	utf-8
1.	x		
2.		x	
3.			x
4.	x	x	
5.	x		x
6.		x	x
7.	x	x	x

De la tabla se desprende que existe la posibilidad de responder con más de una codificación. Esto puede ocasionar confusión en algunos casos (ya que hay dos o más interpretaciones para un mismo dominio), pero garantiza que el cliente al menos va a ver correctamente el dominio en una de las codificaciones.

Cabe destacar que actualmente el *whois* responde con caracteres ISO8859-1, por ejemplo para codificar palabras con acentos como: *Organización*.

## 4.5. Almacenamiento, ¿IDNA, Unicode o ISO8859-1?

Primero se debe determinar si es que el sistema de almacenamiento soporta el formato ISO8859-1 o Unicode, y en que codificación: UTF-8, UTF-16, UTF-32. Además, se debe tomar en cuenta la realidad del sistema actual, como por ejemplo el lenguaje de implementación.

### 4.5.1. Formulario de Dominio

Actualmente en NIC Chile los dominios son administrados utilizando un formulario. Este formulario contiene toda la información relacionada con el dominio, desde los servidores de DNS hasta la información de facturación.

Un formulario se encuentra identificado por la siguiente información, que llamaremos *identificador*:

Cuadro 4.3: Identificador de un Formulario

Estado	Nombre de Dominio	Stamp	Extensión
--------	-------------------	-------	-----------

Para cada identificador, podemos encontrar una serie de campos que forman los encabezados y el contenido del formulario.

Identificador `en-tramite/leyton/20040127160126.dat`  
*(estado/dominio/stamp.extension)*

Encabezados Internos

Encabezados Correo

Campos del Formulario

```

Activado: Tue Jan 27 13:01:26 CLST 2004
Proxpagos: 20031022003412
Papeles: Tue Jan 27 13:01:26 CLST 2004
Tarifa: 0
Notificado: Wed Oct 22 07:24:12 CLST 2003
Modificado: Tue Jan 27 13:02:31 CLST 2004

From: mleyton@dcc.uchile.cl
Reply-To: mleyton@dcc.uchile.cl
To: hostmaster@nic.cl
Organization:
Subject: Modificacion dominio leyton.cl from 172.30.10.62

Version: 2.0
0a. Acción (M=modificacion C=creacion B=borrado): M
0b. Dirección de email para respuesta: mleyton@dcc.uchile.cl
1a. Nombre del dominio: leyton
1b. Nombre o razón social de la organización: Mario Hernan Guillermo Leyton Lueje
1e. RUT de la organización: 13924245-9
1c. Dirección Postal: Manquehue Norte 2329, Vitacura, Santiago
1d. Fecha prevista de comienzo de Operaciones:
2a. Nombre del Contacto Administrativo: Mario Hernan Guillermo Leyton Lueje
2b. Organización del C.A.: Particular
2c. Dirección Postal del C.A.: Manquehue Norte 2329, Vitacura, Santiago
2d. Número Telefónico del C.A.: 2183623
2e. Dirección de correo electrónico del C.A.: mleyton@dcc.uchile.cl
3a. Nombre del Contacto Técnico: Mario Hernan Guillermo Leyton Lueje
3b. Organización del C.T.: Particular
3c. Dirección Postal del C.T.: Manquehue Norte 2329, Vitacura, Santiago
3d. Número Telefónico del C.T.: 2183623
3e. Dirección de correo electrónico del C.T.: mleyton@dcc.uchile.cl
4a. Nombre del servidor primario: ns1.mydomain.com
4b. Dirección IP del servidor primario: 216.34.13.236
5a. Nombre del servidor secundario: ns2.mydomain.com
5b. Dirección IP del servidor secundario: 64.75.34.132
5c. Nombre del servidor secundario: ns3.mydomain.com
5d. Dirección IP del servidor secundario: 64.75.64.140
5e. Nombre del servidor secundario: ns4.mydomain.com
5f. Dirección IP del servidor secundario: 64.75.34.134
6a. Descripción de la organización:
6b. Tipo de organización (e=educación, c=comercial, g=gobierno, m=militar, o=otro): p
6c. NIC Chile actúe como secundario de la zona (S=Si, N=No): N
7a. Otra información:
8a. Forma de pago (F=aviso de cobranza por pagar): F
8b. RUT (para aviso de cobranza): 13924245-9
8c. Nombre o Razón Social (para aviso de cobranza): Mario Hernan Guillermo Leyton Lueje
8d. Domicilio (para aviso de cobranza): Manquehue Norte 2329
8e. Ciudad (para aviso de cobranza): Santiago
8f. Comuna (para aviso de cobranza): Vitacura
8g. Giro Comercial (para aviso de cobranza): Particular
8h. Atención de (para aviso de cobranza): Mario Hernan Guillermo Leyton Lueje
9b. RUT (para emisión de factura): 13924245-9
9c. Nombre o Razón Social (para emisión de factura): Mario Hernan Guillermo Leyton Lueje
9d. Domicilio (para emisión de factura): Manquehue Norte 2329
9e. Ciudad (para emisión de factura): Santiago
9f. Comuna (para emisión de factura): Vitacura
9g. Giro Comercial (para emisión de factura): Particular
9h. Atención de (para emisión de factura): Mario Hernan Guillermo Leyton Lueje
9i. Acepta recibir factura electrónica (S=Si, N=No): N
9j. Correo electronico para recibir factura electronica:

Fin Version 2.0
    
```

Figura 4.1: Ejemplo Formulario

Los formularios han contenido caracteres fuera del ASCII tradicional desde hace un buen tiempo. Para cada dato que el usuario ingresa, es posible contenga caracteres fuera del ASCII tradicional. Sin embargo, al adoptar IDN, algunos campos que tradicionalmente no han contenido caracteres fuera del ASCII podrán hacerlo. Estos campos corresponden a:

- Dominio: 1a



- Emails de contacto: 0b 2e 3e 9j
- Servidores de DNS: 4a 4b 5a 5b

Las dificultades de tener emails de contacto no ASCII se discutieron en la Sección 4.4.2. El caso de los servidores de DNS debe ser tratado con especial cuidado, ya que requiere modificar el proceso de generación de la zona `CL`<sup>2</sup>.

La zona `CL` se genera a intervalos regulares recopilando los servidores de DNS ingresados por los usuarios en los campos del formulario. Con IDN, los servidores pueden contener caracteres no ASCII, y por lo tanto es necesario realizar dos nuevos procesos.

- Validar los servidores al momento de inscribir. Es decir, que puedan ser codificados exitosamente por el algoritmo `ToASCII`. Y adicionalmente, en caso de ser servidores bajo `.CL`, validar que contengan caracteres extendidos adecuados.
- Almacenar la codificación ASCII (el resultado de `ToASCII`) en el formulario, y utilizar este campo para genera la zona `CL`. O alternativamente, almacenar directamente en alguna codificación (ya sea ISO8859-1 o UTF-8), y transformar con `ToASCII` los servidores al momento de generar la zona `CL`.

Los formularios se encuentra almacenados actualmente en dos lugares: File System (`ext3`) y en la Base de Datos (`MySQL`). Esto se debe a que los formularios se encuentran en un proceso de migración desde el File System hacia la Base de Datos.

#### 4.5.2. Sistema de Archivos (*File System*)

Los *File Systems* que nativos soporta Linux hoy en día (`ext2`, `ext3`, `ReiserFS`) son independientes de la codificación utilizada<sup>3</sup>. Por esta razón, la codificación queda en manos de las aplicaciones que escriben al *File System*. Esto puede ocasionar serios conflictos, ya que en un mismo *File System* pueden existir archivos o directorios en diferentes codificaciones.

Migrar un *File System* de una codificación a otra es un tema que escapa del alcance de este documento[5]. Sin embargo, es posible decir que depende de las codificaciones locales, por ejem-

---

<sup>2</sup>La zona corresponde al espacio de nombres de dominio que se administran bajo `.CL`

<sup>3</sup>No así `FAT` o `NTFS`

plo: `localedef -i en_US -f UTF-8 en_US.UTF-8`. Y, de cómo las aplicaciones respeten dichas configuraciones locales al momento de escribir nombres de archivos y su contenido.

Existen dos puntos de interés al implementar IDN en los formularios del *File System*:

1. Para identificar un formulario en el sistema de archivos, es necesario conocer su *identificador*. Con la incorporación de IDN dicho *identificador* puede contener caracteres no ASCII generado problemas con la codificación utilizada en el *File System*.
2. El contenido del formulario también puede contener caracteres no ASCII.

Una posible solución consiste en escribir los dominios en su equivalente ASCII al momento de utilizar el *File System*. Esta es una solución fácil y rápida, pero presenta serios inconvenientes con la usabilidad del sistema. Ya que la presentación en ASCII no siempre es intuitiva para el ser humano (Sección 3.1).

### 4.5.3. Base de Datos (MySQL)

Actualmente la base de datos MySQL soporta Unicode desde la versión 4.1 en su codificación UTF-8<sup>4</sup>[8]. Por esta razón resulta natural utilizar esta codificación en este sistema de almacenamiento, tanto para el *identificador* como para el contenido del formulario.

### 4.5.4. Recomendación

Es esperable que al momento de lanzar IDN el *File System* no siga siendo utilizado y el almacenamiento de los formularios corresponda únicamente a la base de datos. Sin embargo, existen los siguientes factores a considerar:

- El equipo de desarrollo de NIC1 aún no se siente conforme con Unicode en Perl.
- El *File System* debe ser soportado, en caso que exista alguna contingencia que obligue abandonar la Base de Datos.

---

<sup>4</sup>Hasta 3 bytes de largo.

- Al utilizar ACE para el identificador del dominio, no se puede realizar búsquedas por expresiones regulares sin utilizar un índice externo. La mantención de este índice resulta compleja, y puede ser un riesgo mantenerlo actualizado.

Por esta razón, la recomendación de este documento es utilizar en NIC Chile, inicialmente y mientras la tabla de variantes lo permita, la codificación ISO8859-1 para el almacenamiento interno de los formularios.

## 4.6. Modelo Variantes de Caracteres

### 4.6.1. Definición Tabla de Variantes

La tabla de variantes se encuentra definida de la siguiente manera:

- Un número de versión.
- Un número indefinido de filas.
- Dos columnas. La primera indicando el caracter canónico (*CC*) y la segunda indicando las variantes posibles (*VP*) de dicho caracter.

Cuadro 4.4: Definición Tabla de Variantes

Caracter Canónico (CC)	Variantes Posibles (VP)
------------------------	-------------------------

Además la tabla cuenta con ciertas restricciones:

- Los caracteres definidos en la tabla deben ser compatibles con el estándar IDNA.
- En la columna de CC debe existir uno y solamente un caracter.
- En la columna VP puede haber ninguno o muchos caracteres.

- Un caracter puede aparecer una y solamente una vez en la columna CC o VP pero no en la dos, por cada versión de la tabla.

El siguiente extracto muestra los registros más significativos para una posible tabla de variantes:

Cuadro 4.5: Extracto Tabla de Variantes

Caracter Canónico	Variantes Posibles
n	--
ñ	--
a	á
e	é
i	í
o	ó
u	ú ü

En versiones futuras, pueden agregarse caracteres y variantes específicos a otros idiomas. Además, cabe la posibilidad de agregar caracteres especiales como por ejemplo: \$, ©, +, ®. Para generar nuevas versiones de la tabla, las operaciones correspondientes son:

- *Insertar* caracteres nuevos no representa complicaciones, pero debe realizarse con cautela ya que,
- *Eliminar* un caracter de CC o VP solamente puede ser realizado siempre y cuando no existan dominios o variantes asociados a dicho caracter.
- *Mover* un caracter ya insertado en la tabla desde una posición (columna, fila) a otra, corresponde a las operaciones de Eliminación primero e Inserción después, respetando las restricciones del modelo.

#### 4.6.2. Dominio Canónico, Equivalencias y Clases

Hasta ahora, la llave para indexar los dominios en NIC Chile, ha correspondido al nombre de dominio. Con la incorporación del modelo de variantes, este principio ya no es aplicable. Por esta razón es necesario definir el concepto de dominio canónico.

**Dominio Canónico**, corresponde a la traducción de todos los caracteres, de algún dominio o variante, que se encuentran en la tabla VP a su equivalente en VC.

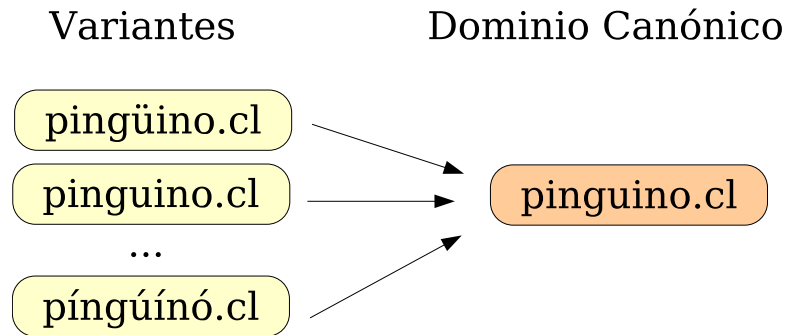


Figura 4.2: Ejemplo de Dominio Canónico

**Identificador**, para cada dominio o variante existe un identificador. Por defecto éste corresponde al dominio canónico pero también puede ser definido explícitamente, en casos excepcionales, como alguna variante de la Clase.

$$Identificador = \begin{cases} Dominio Canonico \\ Variante \end{cases}$$

**Equivalencia Canónica de Dominios**, dos dominios son canónicamente equivalentes si comparten el mismo dominio canónico.

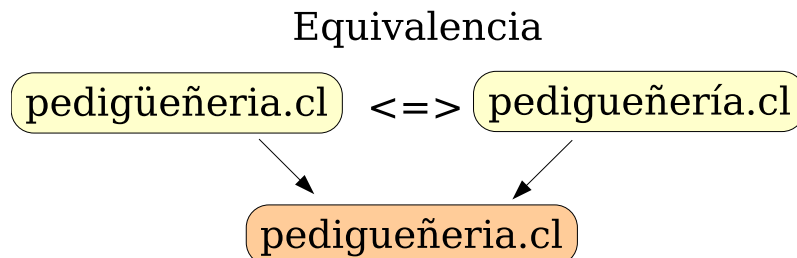


Figura 4.3: Ejemplo de Equivalencia

**Equivalencia de Dominios**, dos dominios son equivalente si comparten el mismo identificador.

**Clase de Dominios**, dos dominios pertenecen a una misma Clase si es que son equivalentes. Por lo general, una Clase esta compuesta por el dominio canónico, y todas las variantes generadas a partir del mismo.

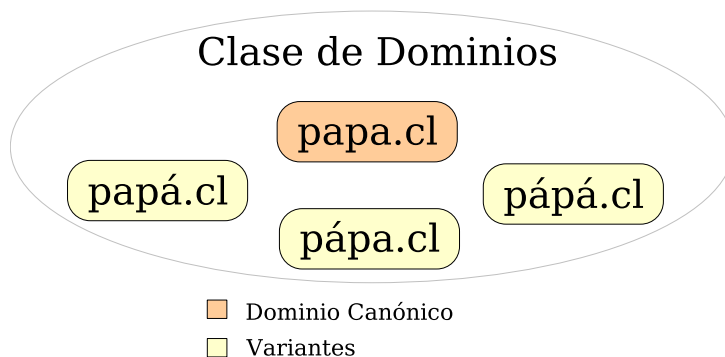


Figura 4.4: Ejemplo de Clase de Dominios

#### 4.6.2.1. Partición Forzada de la Clase

Es posible que en el futuro, por alguna razón de fuerza mayor<sup>5</sup>, sea necesario particionar una Clase de dominios. Para lograr esto, se debe redefinir el identificador de los dominios en cuestión.

**Clase Forzada** corresponde a un subconjunto de variantes de una Clase, se encuentra formada por todos los dominios que son equivalentes y cuyos identificadores no corresponden al dominio canónico.

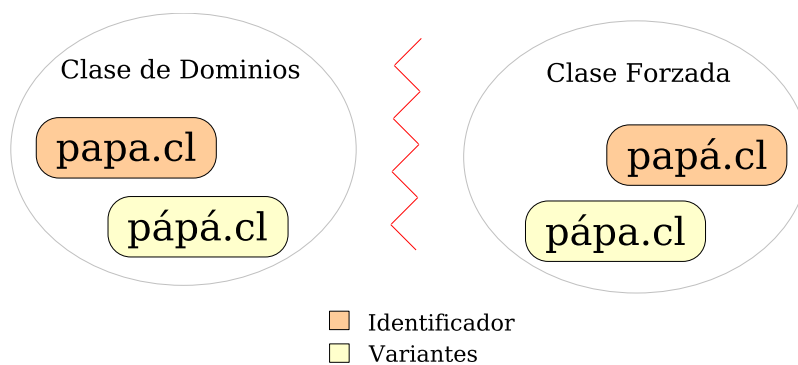


Figura 4.5: Ejemplo de Clase Forzada

<sup>5</sup>Por ejemplo una orden judicial.

Es importante destacar que una Clase puede ser particionada en más de una Clase Forzada, y que cada variante puede existir solamente una vez, ya sea en alguna Clase Forzada o en la Clase original.

Es fácil modificar el identificador para aceptar Clases Forzadas a partir de variantes de diferentes Clases, sin embargo esto no es recomendable.

### 4.6.3. Operaciones

#### 4.6.3.1. Proceso de Inscripción y Asignación

Al momento de solicitar un registro de nombre de dominio, actualmente un cliente puede utilizar solamente LDH<sup>6</sup>. Con la incorporación de IDN el usuario podría solicitar dominios con caracteres definidos en la Tabla de Variantes. Este dominio correspondería entonces al *dominio inicial*, y se reservarían a partir de éste, una serie de variantes.

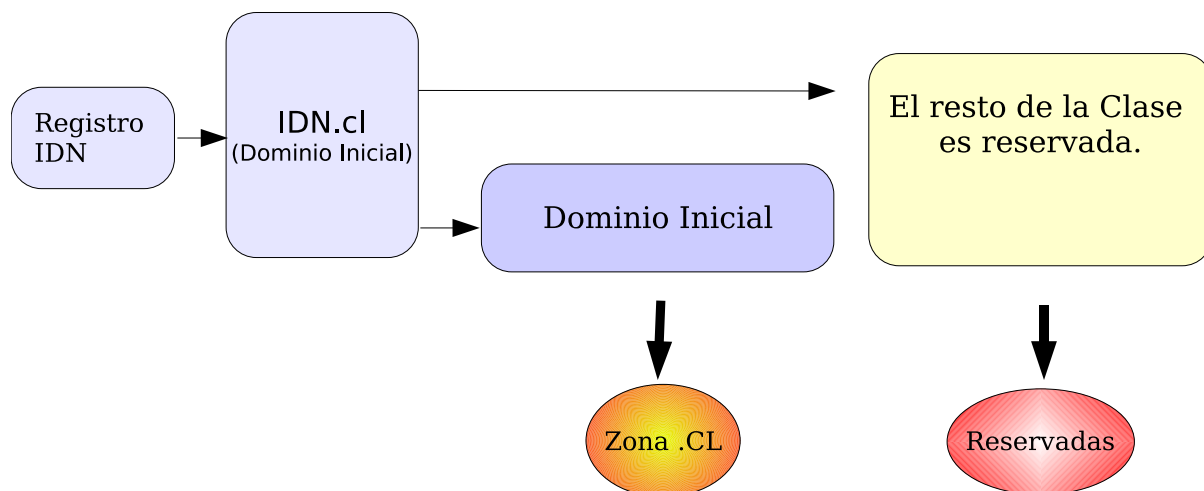


Figura 4.6: Proceso de Inscripción con Variantes

**Dominio Inicial** Corresponde a la variante elegida inicialmente por el cliente para realizar la inscripción. Puede corresponder al dominio canónico o alguna variante.

Una vez asignado el *dominio inicial*, el cliente podría activar las variantes que estime convenientes mediante una tarifa. Es decir, la asignación del *dominio inicial* corresponde a la asignación de la

<sup>6</sup>Del inglés, Letters Digits Hyphens (-)

Clase a la que pertenece.

La asignación de una Clase se mantendría vigente mientras no se desactiven todas las variantes de la Clase, y pasado un período de gracia definido por NIC Chile.

A partir de este momento se hablará indistintamente entre variantes y dominios, excepto cuando se indique lo contrario.

Cabe destacar que el largo máximo de un dominio tradicional es de 63 caracteres (sin incluir el .cl). Con IDN este largo disminuye arbitrariamente. Ahora la restricción de largo dependerá del éxito al momento de codificar el dominio mediante el algoritmo ToASCII (Sección 3.1.3).

#### **4.6.3.2. Renovaciones**

Los procesos de renovaciones de las variantes de una Clase operarían de manera independiente, de acuerdo con el régimen regular. Los plazos de renovación para cada variante de una Clase podrían ser distintos, dependiendo del plazo solicitado por el asignatario de la Clase al momento de activar la variante.

#### **4.6.3.3. Conflictos y Revocaciones**

Al incorporar modelo de variantes es posible generar un conflicto por dominios distintos, pero que son equivalentes. Es decir, variantes de un mismo dominio canónico, o dominios pertenecientes a una misma Clase.

Por esta misma razón, las revocaciones se relizarían entre dominios equivalentes. Es decir, al revocar un dominio canónico o alguna de sus variantes, se revocaría toda la asignación de la Clase a la que pertenece el dominio.

#### **4.6.3.4. Proceso de Activación de Variantes**

La activación de variantes solamente podría ser realizada por el asignatario de la Clase. Las posibles variantes se encuentra definidas a partir de la Tabla de Variantes. Una vez recibida la notificación del pago, se procedería a la activación de la variante.



El proceso de activación de variantes no tendría que publicar la solicitud en el listado de *entrámite*.

#### **4.6.3.5. Desactivaciones**

Las desactivaciones de cada variante estarían asociadas a su propio proceso de renovación regular, sin afectar el estado de activación de otras variantes pertenecientes a la misma Clase.

#### **4.6.3.6. Transferencias**

Al transferir la asignación de alguna variante, se transferiría la asignación de la Clase a la que pertenece. No es posible transferir solamente alguna(s) variante(s) de la Clase.

#### **4.6.3.7. Eliminaciones**

Las eliminaciones de variantes serían directas y sin inconvenientes. Eliminar una variante no afectaría la asignación del dominio inicial o el resto de las variantes de la Clase.

#### **4.6.3.8. Modificaciones**

Las variantes de una misma Clase serían afectadas por cualquier modificación realizada a algún miembro de la Clase. Esto incluye datos del formulario como por ejemplo: correos de contacto, datos de facturación y servidores de DNS, entre otros. No se descarta la posibilidad de independizar algunos de estos datos en el futuro.

### **4.6.4. Costos y Tarifas**

Como fue discutido con anterioridad (sección 2.4), existen varios costos asociados al modelo de variantes de caracteres.

- *¿Cuál es el costo de reservar los dominios con variantes?* Este costo corresponde al costo de oportunidad de inscribir un dominio. Para una Clase de dominio, el costo de reservar dicha Clase corresponde a:

$$Costo Clase = \sum_i \Pr(\text{inscripcion de } d_i) \cdot Tarifa Dominio \quad / \forall d_i \in Clase$$

- *¿Debe este costo estar incluido en el registro del dominio?* En este modelo se ha dispuesto no incluir este costo en el registro del dominio base.
- *¿Cuál debe ser la tarifa de activación de dominios con variantes?* Una posible tarifa de activación podría estar definida por la esperanza:

$$E(\text{variante}) = \frac{Costo Clase}{n}, \quad / n = | Clase |$$

Sin embargo, podemos suponer que la varianza es muy grande ya que las variantes son muy codiciables o no deseables. Es decir,  $\Pr(d_i) \approx 1 \vee \Pr(d_i) \approx 0$ . Bajo este supuesto, la esperanza queda invalidada como un buen estimador para la tarifa. Una alternativa podría suponer el número de variantes codiciadas  $E(\text{numcod})$ , y calcular la esperanza en base a esto:

$$E'(\text{variante}) = \frac{Costo Clase}{E(\text{numcod})}$$

$$E'(\text{variante}) = \frac{Tarifa Dominio \cdot \sum \Pr(d_i)}{E(\text{numcod})}$$

$$E'(\text{variante}) \approx \frac{Tarifa Dominio \cdot E(\text{numcod})}{E(\text{numcod})}$$

$$E'(\text{variante}) \approx Tarifa Dominio$$

Por lo tanto es razonable definir una tarifa activación para las variantes, igual a la tarifa de inscripción de un dominio.

## 4.7. *Sunrise* con Variantes

Es posible modificar el Modelo de Variantes para aplicarlo solamente durante el *sunrise*. Para esto sería fundamental manejar las solicitudes como independientes como se recomienda en la Sección 4.6. El proceso de operaciones entonces quedaría descrito en la siguiente figura.

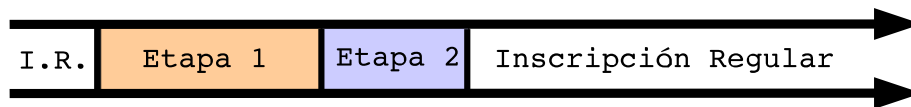


Figura 4.7: Línea de tiempo para *sunrise*.

### 4.7.1. Primera Etapa

En la primera etapa, solamente el asignatario de una Clase podría inscribir variantes. Para esto se define una Tabla de Variantes como se discutió en la Sección 4.6. Luego, existirían dos maneras de permitir el registro de variantes:

- El registro de las variantes se realizaría utilizando el formulario tradicional de inscripción en la página web. Cualquier persona podría realizar la inscripción sin verificar su identidad. Al momento de proceder con la inscripción, si el dominio corresponde a una variante de una Clase asignada, y el RUT especificado en la variante concuerda con el titular de la Clase, se permitiría la inscripción de la variante.
- Solamente alguno de los contactos autorizados para la Clase podría realizar la activación de una variante. Para esto se proveería una interfaz especial, donde se valida al usuario utilizando un código de autorización. Los campos del formulario podrían ser poblados automáticamente por NIC Chile en base al dominio inicial.

Es importante destacar que en ninguno de los casos, la variante debe pasar por un período *en-tramite*, propiamente tal. Ya que al utilizar *sunrise* se determinan *a priori* los derechos de asignación. Esto podría generar problemas durante el *sunrise* cuando existe más de una persona que cree tener derechos sobre el dominio. Casos como estos podrían ser derimidos después del *sunrise* mediante un proceso de revocación.

La recomendación entre las dos alternativas es optar por la número dos. La principal ventaja sería que la variante es activada por una persona autorizada, disminuyendo la posibilidad de registros malintencionados.

#### **4.7.1.1. Eliminaciones, Revocaciones y Transferencias**

Durante este período sería posible que cambie el asignatario de una Clase, ya sea porque se revoca el dominio, se transfiere o se elimina. Existen dos maneras de manejar esta situación, a partir de este momento se hablará solamente de transferencia, considerando similares los casos de revocación o eliminación:

- Al transferir el *identificador* o alguna variante, se transferiría la Clase completa, es decir todas sus variantes.
- Al transferir el *identificador* no se transferirían las variantes ya activadas. Una vez transferido el *identificador*, el nuevo asignatario podría activar las variantes que aún queden libres bajo su titularidad. En caso de transferir una variante que no corresponda al *identificador*, el nuevo asignatario no podría generar nuevas variantes.

Mientras la primera alternativa intenta abordar esta etapa como una implementación del Modelo de Variantes, la segunda solamente lo hace protegiendo el proceso de inscripción. Dado que esta etapa corresponde a un período transitorio dentro del *sunrise*, parece más natural optar por la segunda alternativa.

#### **4.7.2. Segunda Etapa (Transición)**


Esta etapa consiste en un período de transición entre el Modelo de Variantes y el Modelo de Inscripción Regular. Este período es relativamente breve, y su principal objetivo es esperar un período razonable de tiempo para la notificación de los pagos *offline* (Servipag).

Durante esta etapa no sería posible registrar o activar variantes. El resto de las operaciones no presentarían restricciones o alteraciones.

Al concluir esta etapa se descartarían todas las solicitudes por variantes que figuren impagas.

### 4.7.3. Tercera Etapa (Inscripción Regular)

A partir de este momento se redefiniría la Tabla de Variantes dejando todos los caracteres en la columna de caracteres canónicos:



Caracter Canónico	Variantes Posibles
n	--
ñ	--
a	á
e	é
i	í
o	ó
u	ú ü

Caracter Canónico	Variantes Posibles
n	--
ñ	--
a	--
á	--
e	--
é	--
i	--
í	--
o	--
ó	--
u	--
ú	--
ü	--

Figura 4.8: Modificación Tabla de Variantes

Al redefinir la Tabla de Variantes, las Clases pasan a ser de tamaño uno, es decir, cada dominio es independiente. Por lo tanto, ya no se podrían activar variantes porque no existen. Entonces, el único proceso de inscripción habilitado correspondería al regular.

También sería posible solicitar revocaciones de variantes activadas durante la primera etapa, sin que esto signifique una revocación de la Clase completa.

## 4.8. Modelo Biblioteca NIC::IDN

### 4.8.1. Descripción

A pesar que existe una biblioteca con excelentes cualidades como LibIDN, aún persisten dos grandes problemas por resolver.

- Inmadurez de la biblioteca LibIDN genera, en ocasiones, cambios sustanciales en la forma de operar entre un *release* y otro.

- No satisface las necesidades del Modelo de Variantes.

Por esta razón es necesario encapsular la biblioteca en otra. Una específica a las necesidades de NIC Chile, desde donde se pueda extender fácilmente las funcionalidades protegiendo las aplicaciones clientes.

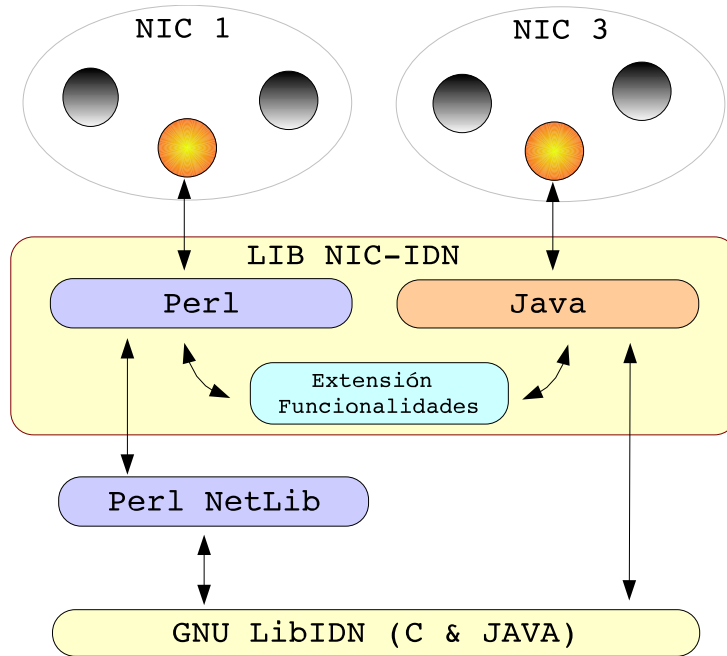


Figura 4.9: Biblioteca NIC-IDN

#### 4.8.2. Requerimientos Funcionalidades Básicas

Por funcionalidades básicas, hacemos referencia al encapsulamiento del estándar IDN implementado en la biblioteca LibIDN. Las funcionalidades corresponden a:

- Algoritmo ToASCII (Sección 3.1.3)
- Algoritmo ToUnicode (Sección 3.1.4)

### **4.8.3. Requerimientos Funcionalidades Extendidas (Modelo Variantes de Caracteres)**

Por funcionalidades extendidas, hacemos referencia a características específicas de la biblioteca NIC::IDN. Principalmente corresponden a herramientas para operar con el modelo de Variantes de Caracteres. Estas funcionalidades son:

1. Para algún dominio o variante, determinar si dominio es válido de acuerdo a la Tabla de Variantes.
2. Para algún dominio o variante, obtener su representación canónica.
3. Para algún dominio o variante, obtener su identificador.
4. Determinar si dos dominios o variantes pertenecen a la misma Clase.
5. Comparación de equivalencia canónica para dominios y variantes.
6. Comparación de equivalencia para dominios y variantes.
7. Para algún dominio o variante, obtener la Clase a la que pertenece.
8. Para algún dominio o variante, determinar si pertenece a una Clase Forzada.

El API relacionado con estas funcionalidades se encuentra descrito en el Apéndice B.

# Capítulo 5

## Implementación y Testing Biblioteca NIC::IDN

### 5.1. Introducción

Esta biblioteca tiene como objetivo servir de base para la implementación de IDN en la plataforma operacional de NIC Chile. El diseño de la biblioteca fue descrito en el capítulo anterior, y considera una posible adopción del Modelo de Variantes. El API de la biblioteca se encuentra en el Anexo B. En este capítulo, también se describen las herramientas de validación de la biblioteca.

### 5.2. Metodología

La metodología utilizada durante el desarrollo de esta memoria corresponde a *Painless* (indolora) [17]. Esta metodología permite mantener un desarrollo ágil de software dividiendo el trabajo completo en pequeñas tareas. A cada tarea se asigna inicialmente un *tiempo estimado*. A medida que se realiza la tarea se actualiza el *tiempo transcurrido*, y el *tiempo restante* que falta para el término de la tarea. La tarea es finalizada cuando el tiempo restante llega a cero.

Cuando alguna tarea es finalizada, se calcula la eficiencia como el promedio de:  $\frac{\text{tiempo estimado}}{\text{tiempo transcurrido}}$  para todas las tareas terminadas. Este factor se utiliza para estimar los recursos (tiempos) necesarios para concluir las tareas aún no finalizadas del proyecto.



Las principales virtudes de *Painless* corresponden a: su sencillez de administración, facilidad para visualizar el estado del trabajo, y capacidad de estimar los tiempos necesarios para finalizar el proyecto.

### 5.3. Implementación del Módulo Perl `NIC::IDN.pm`

El módulo `NIC::IDN.pm` se encuentra escrito en Perl. Además de operar como *wrapper* para las funcionalidades con la biblioteca GNU LibIDN, agrega funcionalidades extendidas relacionadas con el Modelo de Variantes. Por esta razón, esta biblioteca administra sus propias tablas.

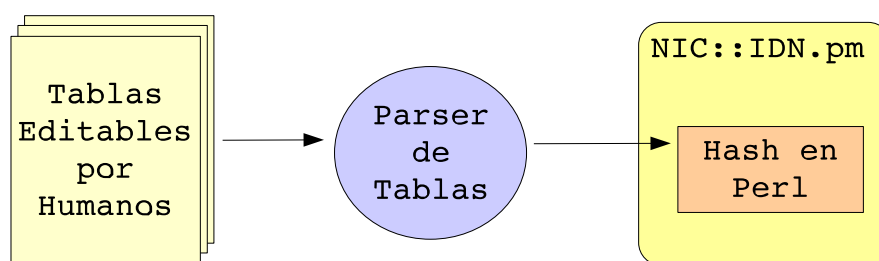


Figura 5.1: Transformación de Tablas de Caracteres

Estas tablas son escritas en archivos de texto plano, con un formato editable por una persona. Para incluirlas en el módulo son parseadas en varios *hashes* de Perl. Finalmente, estos *hashes* son incluidos y utilizados por el módulo de Perl.

La principal ventaja de utilizar este esquema, es que el trabajo de parseo se realiza solamente una vez *off-line*, y no cada vez que se utiliza el módulo. Por otro lado, la desventaja es que las tablas deben ser regeneradas cada vez que se modifica o agrega una nueva tabla. Sin embargo esto no es un problema, ya que es esperable que las modificaciones de las tablas sean escasas.

#### 5.3.1. Tablas Editables

Como fue discutido en la Sección 4.6.1, una tabla se encuentra definida por un TLD y una versión:

```
version "1.0"
```

```
tld cl
```

Ademas, está compuesta por una serie de registros en forma de filas. Para cada fila, el primer *code point*<sup>1</sup> corresponde al caracter canónico. Todo lo que se encuentra despues del “|” corresponde a las variantes. Si existe más de una variante, éstas se encuentran separadas por el caracter “:”. Las líneas que comienzan con el caracter “#” corresponden a comentarios.

```
# a | LATIN SMALL LETTER A WITH ACUTE
U+0061|U+00E1
# e | LATIN SMALL LETTER E WITH ACUTE
U+0065|U+00E9
# i | LATIN SMALL LETTER I WITH ACUTE
U+0069|U+00ED
# o | LATIN SMALL LETTER O WITH ACUTE
U+006F|U+00F3
# u | LATIN SMALL LETTER U WITH ACUTE : LATIN SMALL LETTER U WITH DIAERESIS
U+0075|U+00FA:U+00FC
# n | LATIN SMALL LETTER N WITH TILDE
U+006E|U+00F1
```

Los *code points* se encuentran descritos en hexadecimal, y corresponden al caracter en minúsculas. Esto último se debe a que el resultado final de la normalización *nameprep* entrega los caracteres en minúsculas.

### 5.3.2. Hash Canónico->Variantes

Este tipo de *hash* es uno de los resultados de *parsear* la tabla. Para cada TLD, y cada versión de la tabla se puede obtener un *hash* de este tipo.

Utilizando este *hash* es posible obtener, para algún caracter canónico, un arreglo de caracteres con sus variantes. El método interno de la biblioteca que permite obtener el *hash* es:

```
sub getVariantsHash $tld $version
```

El resultado corresponde a un *hash* descrito por:

---

<sup>1</sup>En este caso corresponde a la enumeración Unicode del caracter.

**Llave** Algún caracter canónico definido en la tabla.

**Valor** Un arreglo que contiene las variantes de la llave.

A continuación se muestra la función que genera recursivamente todas las variantes posibles de un dominio:

```
sub getClaseRec{

    my $domain = shift;    #Dominio generado hasta ahora
    my $version = shift;  #Version de la tabla a utilizar
    my $offset = shift;   #offset del dominio

    my @newelements=();

    #Condicion de borde para la recursividad
    if ($offset >= length($domain) || substr($domain,$offset,1) eq "."){
        push @newelements, $domain;
        return \@newelements;
    }

    #Recuperamos el hash
    my %vhash = %{getVariantsHash(getTLD($domain), $version)};

    #Utilizamos el hash para obtener un arreglo
    #con las variantes del caracter
    my $c=substr($domain,$offset,1);
    my @variants= @{$vhash{ord($c)}};

    #Recursivamente generamos las variantes
    push @newelements, @{getClaseRec($domain, $version,$offset+1)};
    for(my $i=0; $i < scalar(@variants); $i++){
        substr($domain, $offset, 1, chr($variants[$i]));
        push @newelements,
            @{getClaseRec($domain, $version,$offset+1)};
    }

    return \@newelements;
}
```

### 5.3.3. Hash Variante->Canónico

Existe un *hash* de este tipo para cada TLD y cada versión de la tabla. Este *hash* permite obtener el caracter canónico para cualquier variante habilitada. El método interno de la biblioteca que permite recuperar el *hash* es:

```
sub getCanonizingHash $tld $version
```

El resultado de invocar esta función es un *hash* donde:

**Llave** Corresponde a la variante.

**Valor** Corresponde al caracter canónico asociado a la llave.

El siguiente ejemplo muestra como se puede convertir un dominio en su representación canónica utilizando este *hash*.

```
sub getCanonicoVersion{
    my $domain= shift;      #Dominio a transformar
    my $version = shift;    #Version tabla a utilizar
    my $encoding=shift;     #Codificacion ISO8859-1 o UTF-8

    #Aplicamos nameprep para normalizar el dominio
    my $domain_np=idn_prep_name($domain, $encoding);

    #Obtenemos el TLD del dominio
    my $domain_tld=getTLD($domain_np);

    #Recuperamos el hash para este tld-version
    my %chash=%{getCanonizingHash($domain_tld,$version)};

    #Cada caracter es transofrmado a su canonico.
    #Si el canonico no existe, se utiliza el mismo caracter.
    my $canonico;
    for(my $i=0; $i < length($domain_np); $i++){
        my $c=substr($domain_np,$i,1);
```

```

    if (defined $chash{ord($c)}){
        $canonico.=chr($chash{ord($c)});
    }
    else{
        $canonico.=$c;
    }
}
return $canonico;
}

```

## 5.4. Testing de la Biblioteca

Para poder verificar la correcta operación de la biblioteca, se han diseñado una serie de pruebas que permiten validar la correctitud operacional. A continuación se muestran los resultados de los test automáticos realizados sobre la biblioteca.

```

Test 1 Canonico con UTF-8      ... OK
Test 2 Canonico con ISO8859-1  ... OK
Test 3 Obtener Identificador con UTF-8 ... OK
Test 4 Obtener Identificador con ISO8859-1 ... OK
Test 5 Igualdad Canonica con UTF-8 ... OK
Test 6 Igualdad Canonica con ISO8859-1 ... OK
Test 7 Igualdad con UTF-8      ... OK
Test 8 Igualdad con ISO8859-1  ... OK
Test 9 Misma clase con UTF-8   ... OK
Test 10 Misma clase con ISO8859-1 ... OK
Test 11 Dominio es valido con UTF-8 ... OK
Test 12 Dominio es valido con ISO8859-1 ... OK
Test 13 Pertenece a una Clase Forzada con UTF-8 ... OK
Test 14 Pertenece a una Clase Forzada con ISO8859-1 ... OK
Test 15 Generacion de clases con UTF-8 ... OK
Test 16 Generacion de clase con ISO8859-1 ... OK
Test 17 Es un dominio IDN? con ISO8859-1 ...OK
Test 18 Es un dominio IDN? con UTF-8 ...OK

```

# Capítulo 6

## Implementación y Testing, Adopción de IDN en la plataforma NIC1

### 6.1. Introducción

Como fue discutido en el capítulo de diseño, si se desea aplicar un período de *sunrise*, el Modelo de Variantes, ambos o ninguno de los dos, es necesario preparar la plataforma NIC1 para trabajar con caracteres extendidos. En este capítulo se describen los procedimientos realizados para lograr esta preparación. Para esto fue fundamental contar con un ambiente de desarrollo idéntico al de producción. Este ambiente fue provisto por NIC Chile.

En primer lugar se desarrollaron *tests* sobre las funciones modificadas (o agregadas) de los distintos módulos de la plataforma. Estos *tests* pueden ser considerados de *caja negra* ya que están formados por un conjunto de pruebas asociadas a un resultado esperado.

Luego se verificaron las operaciones de la plataforma. Para cada operación, se verificó que funcionara de manera adecuada, y en los casos que no fue así se corrigieron los errores hasta lograr los resultados deseados.

Finalmente se realizaron pruebas y verificaciones misceláneas.

## 6.2. Módulos NIC1

En esta etapa, los *tests* y la implementación van mano en mano. Es decir que, para cada función que se desea modificar o agregar, primero se implementa un *test* de verificación. Una vez que el *test* es aprobado, se procede a modificar la función. La función queda correctamente modificada una vez que apruebe los *tests* ampliados. A continuación se detalla la metodología:

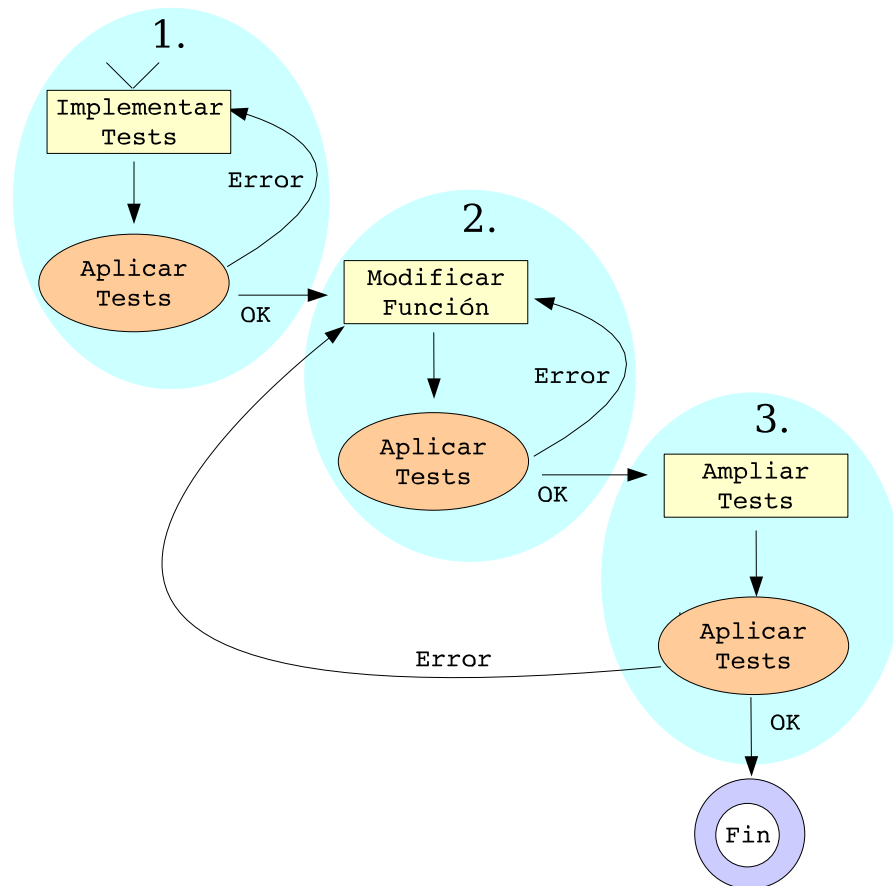


Figura 6.1: Metodología Modificación de Funciones

1. Inicialmente se implementa un conjunto de *tests* que verifican la correcta operación de la función antes de ser modificada.
2. Luego se modifica la función. Una vez modificada se aplican los *tests*, en caso de ser aprobados la función aún cuenta con las propiedades iniciales.
3. Finalmente se amplían los *tests*, agregando por ejemplo, casos nuevos relacionados con la funcionalidad deseada. Una vez que se aprueben todos los *tests* tenemos algún grado de confianza que la función se comporta adecuadamente.

En general los *tests* son automatizados y ejecutados para validar el correcto funcionamiento de los módulos.

### 6.2.1. Form

**mal\_formado** Verifica que un dominio contenga los caracteres adecuados. Fue modificada para ser *wrapper* de `Util::mal_formado`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

**muy\_largo** Verifica que el largo de un dominio sea aceptable en el DNS. Fue modificada para ser *wrapper* de `Util::muy_largo`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

**mal\_email** Verifica que un correo tenga la forma correcta. Fue modificada para ser *wrapper* de `Util::verificar_email`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

En las siguientes funciones se agregó normalización al parámetro `$dominio`: *List\_stamps*, *Find\_dir*, *Find\_newdir*, *Find\_form*, *locateForm*, *locateState*, *Find\_form\_lock*, *write*, *Mv\_form*, *hayTransferencia*, *setTransferencia*, *releaseTransferencia*.

### 6.2.2. Ingresar Solicitud

**mal\_formado** Verifica que un dominio contenga los caracteres adecuados. Fue modificada para ser *wrapper* de `Util::mal_formado`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

**muy\_largo**, Verifica que el largo de un dominio sea aceptable en el DNS. Fue modificada para ser *wrapper* de `Util::muy_largo`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

Se modificaron las llamadas a la función `lowercase` para que ahora utilicen la función de normalización `NIC::IDN::idn_prep_name()`. Además se agregó la normalización al parámetro `$dominio` en las siguientes funciones: *prepare\_mail*, *print\_formulario*, *dominio\_prohibido*.



### 6.2.3. Receive Mail

No fue necesario modificar este módulo ya que hace sus verificaciones a través del API de Form.

### 6.2.4. CLIO

**activa** Para cada formulario, esta función se encarga de generar los archivos de zona necesario. Estos pueden ser parte de: `cl.zone`, `named.conf` en el servidor primario `ns.nic.cl`. O también parte de los archivos `named.conf` y `dominio.cl.zone` en `secundario.nic.cl`. Fue necesario modificar esta función para que escriba los archivos de configuración con los nombres de dominio transformados en ASCII.

### 6.2.5. Dom-CL

**mal\_formado** Verifica que un dominio contenga los caracteres adecuados. Fue modificada para ser *wrapper* de `Util::mal_formado`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

### 6.2.6. Show Form

No fue necesario modificar este módulo.

### 6.2.7. Ultimos

**mal\_formado** Verifica que un dominio contenga los caracteres adecuados. Fue modificada para ser *wrapper* de `Util::mal_formado`. Su estado actual es *deprecated*, se recomienda utilizar directamente la función en el módulo Util.

## 6.2.8. Util

**mal\_formado** Verifica que un dominio contenga los caracteres adecuados. Fue modificada para aceptar caracteres IDN utilizando las tablas de la biblioteca `NIC : : IDN`.

**muy\_largo**, Verifica que el largo de un dominio sea aceptable en el DNS, es decir, menor o igual a 63 caracteres. Al codificar caracteres con *punycode* el tamaño del dominio aumenta, por lo que se modificó la función para tomar esta consideración.

**verificar\_email** Se encarga de verificar la sintaxis de un email. Fue modificado para no aceptar emails con caracteres extendidos o codificaciones *punycode*.

**chek\_hostname** Verifica que este bien escrito el nombre de un servidor. Fue modificado para verificar que el servidor ingresado pueda ser transformado éxistosamente con el algoritmo ToASCII. Esto significa que ahora se pueden aceptar dominios con IDN en el campo de servidores del formulario.

## 6.2.9. Zone Grep y LogGrep

Fue necesario modificar estos módulos para que realicen búsquedas utilizando parámetros no ASCII. La verificación de estos modulos se realiza manualmente a través de la página web.

## 6.3. Operaciones

Las operaciones están relacionadas con el proceso de registro de nombres de dominio. Algunas son representadas mediante formularios. Es decir que, para representar una operación se crea un formulario asociado. Además, existen operaciones que se realizan sobre formularios ya creados.

### 6.3.1. Formularios de Operaciones

Estos *test* son realizados a través de la interfaz web. Por ahora, son realizados manualmente, pero en el futuro se espera contar con una implementación automatizada.

1. Creación

2. Renovación
3. Modificación
4. Transferencia
5. Revocación
6. Eliminación

## 6.3.2. Operaciones sobre Formularios

Para la mayoría de las operaciones sobre formularios se implementaron verificaciones automáticas utilizando un formulario creado específicamente con este propósito.

### 6.3.2.1. Activar

Se agregó el *flag* “-q” de *quick* (no hace la consulta dns). Además se corrigió un *bug* que afectaba dominios en estado *en-tramite*. El *test* verifica que el comando activa agregue el encabezado Z y elimine el encabezado D, además verifica que el dominio sea incluido en `DIRTY_doms` para que pueda ser agregado a la zona *CL* por el módulo *CLIO*.

Las verificaciones que se deben realizar para este comando son:

1. Dominio sin requisitos (pago, papeles) no es activado.
2. Dominio sin requisitos es activado con *flag* “-f” (forzado).
3. Dominio con requisitos es activado sin necesidad de utilizar el *flag* “-f”.
4. Dominio previamente activado no es activado.
5. Dominio ya activado es activado solo con *flag* “-f” .
6. Dominio desactivado no es activado.
7. Dominio desactivado es activado solamente con *flag* “-r” (reactiva).
8. Formulario de revocación no se puede activar, incluso con *flag* “-f”.

Actualmente se verifican la condiciones: 3, 6, 7 y 8.

#### **6.3.2.2. Desactivar**

El *test* verifica que el comando desactiva elimine el encabezado Z y agregue el encabezado D. Además verifica que el dominio sea incluido en DIRTY\_doms para ser eliminado de la zona CL en la próxima actualización.

Las verificaciones que se deben realizar para este comando son:

1. Dominio ya desactivado no es vuelto a desactivar.
2. Dominio no activado no se puede desactivar.
3. Cuando existe una revocación se desactiva el dominio original y no la revocación.
4. Dominio activado es desactivado.

Actualmente se verifican las condiciones: 1, 3 y 4.

#### **6.3.2.3. Asignar**

El *test* verifica que un formulario de dominio cambie de estado *en-tramite* a *asignado*.

Las verificaciones que se deben realizar para este comando son:

1. Las creaciones, modificaciones y renovaciones son asignadas.
2. Si existe un conflicto no se asigna la creación.
3. Una revocación no puede ser asignada a menos que se especifique el stamp explícitamente.
4. Dominio congelado no se asigna.

Actualmente se verifican la condiciones: 1, 2 y 4.

#### **6.3.2.4. Borrar y Desistir**

Cuando un dominio es eliminado su estado cambia. Si el dominio se encontraba en estado *asignado* entonces pasa a *asignado-old*, si su estado corresponde a *en-tramite* entonces su estado pasa a *en-tramite-old*.

1. Dominio asignado es borrado y pasa estado *asignado-old*.
2. Dominio *en-tramite* es desistido y pasa a estado *en-tramite-old*.
3. Dominio en renovación es desistido, y además se borra el formulario de renovación.
4. Dominio *en-tramite* con requisitos solamente es desistido si se utiliza opción “-f”.
5. Si el dominio se encuentra activado, entonces es desactivado al llamar a borra o desiste.
6. Al desistir una transferencia donde el asignado es comodín se borra también el formulario asignado.
7. Cuando se desiste transferencia que reemplaza una renovación, se debe revivir la renovación.

Actualmente se verifican las condiciones: 1, 2 y 3.

#### **6.3.2.5. Modificar**

Las verificaciones que se deben realizar para este comando son:

1. Los campos del dominio puedan ser modificados y que estos cambios se guarden.

Actualmente se verifica la condición: 1.

#### **6.3.2.6. Pagar**

Cuando se paga un dominio, el formulario asociado es marcado con los encabezados respectivos.

1. Un dominio impago es pagado, entonces se marca como pagado.
2. Un dominio ya pagado no se puede volver a pagar.

Actualmente se verifica la condición: 1.

#### **6.3.2.7. Facturar**

Cuando un dominio se factura se agregan los encabezados necesarios al formulario respectivo.

1. Cuando un dominio es marcado como facturado se verifica que tenga el encabezado.
2. Cuando un dominio ya se encuentra facturado se verifica que no se vuelva a facturar.

Actualmente se verifica la condición: 1.

#### **6.3.2.8. Congelar**

Al congelar un dominio se agrega un encabezado que inhibe algunas operaciones sobre el dominio. La verificación consiste en revisar la existencia o inexistencia del encabezado despues de llamar al comando.

Las verificaciones que se deben realizar para este comando son:

1. Dominio es congelado con el comando.
2. Dominio congelado es descongelado al ejecutar el comando.
3. Dominio congelado con *flag* -a (apitutado), no se puede descongelar sin utilizar el *flag*.

Actualmente se verifican la condiciones: 1 y 2.

### 6.3.2.9. Proteger

Al proteger un dominio se agrega al listado `~clcorreo/src/doms`. Los dominios en este listado nunca cambian su estado.

Las verificaciones que se deben realizar para este comando son:

1. Dominio es protegido con el comando.
2. Dominio protegido es desprotegido al ejecutar el comando.
3. Dominio protegido con *flag* -a (apitutado), no se puede desproteger sin utilizar el *flag*.

Actualmente se verifican las condiciones: 1 y 2.

## 6.4. Misceláneos

Estos *tests* no corresponden a otras categorías, o fueron solicitados explícitamente por el equipo de desarrollo de NIC Chile. Por lo general son difíciles de verificar de manera automatizada.

1. *MyMhonarc*. Corresponde al sistema que almacena los correos asociados con los dominios, y que está diseñado para ser visto a través de una página web. El *test* consistió en verificar que el sistema registrara los correos de dominios con caracteres extendidos. Ahora utiliza `Util::mal_formado()` para verificar el dominio. También, se identificó que existe un problema con la base de datos *mysql*, ya que en la configuración actual el *mysql* interpreta `ó==o`. Se corrigió modificando la opción *collation* de la base de datos.
2. *Nómina Correo de Renovación*. Corresponde a un listado enviado mensualmente a una empresa externa para generar y enviar las notificaciones de renovaciones. Este listado corresponde a un archivo de texto plano. El *test* consiste en verificar que se envíe el listado correctamente y corroborar la generación correcta de dominios con caracteres extendidos. Esto se hace actualmente para los dominios tradicionales utilizando verificación manual sobre una muestra de los cupones generados por la empresa externa.

3. *Generar Colilla de Pago.* Corresponde a un cupón PDF que pueden bajar los usuarios desde el sitio web de NIC Chile para cancelar la creación o renovación de un dominio mediante Servipag. El *test* consistió en verificar que se generara correctamente la colilla de pago para dominios con caracteres extendidos. Para esto fue necesario corregir el `carro` de compras para que utilice la función `Util::mal_formado`. Además se agregó el dominio codificado en ASCII al `hash %vars` para poder generar una URL en ASCII desde la planilla (`servipag-unico.html`). Por esta razón, también fue necesario modificar el módulo `boleta` para transformar el dominio codificado en ASCII antes de generar el cupón.
4. *Generar Facturas.* Corresponde a un documento tributario electrónico, y es generado como archivo PDF. Los usuarios pueden bajar la factura desde el sitio web de NIC Chile, o recibirla a través de correo electrónico. El *test* consistió en verificar que la factura se generara correctamente.
5. *Servidor Whois.* Corresponde a una herramienta de consulta que provee información relacionada con algún dominio. Ahora el *whois* acepta consultas por dominios codificados con caracteres extendidos utilizando ISO8859-1<sup>1</sup>. Estas consultas pueden venir con caracteres extendidos o en su representación ACE. En la respuesta del servidor, ahora se entrega el dominio codificado en ISO8859-1 y también en su representación ACE. Además fue necesario modificar el servidor `whoisd` para verificar la sintaxis del dominio utilizando la función: `Util::mal_formado`. El *test* consistió en generar una verificación automática que cumpla con las funcionalidades anteriores.
6. *Sistema de Conflictos.* Corresponde a un sistema que permite manejar los arbitrajes de nombres de dominio. Su verificación y compatibilidad con IDN se encuentra actualmente a cargo del grupo de Ingeniería de NIC Chile.

---

<sup>1</sup>Como fue discutido con anterioridad, el protocolo *whois* no presenta funcionalidades de internacionalización y debe ser supuesto a priori la codificación de los clientes.



# Capítulo 7

## Conclusiones y Trabajo Futuro

### 7.1. Conclusiones

En los primeros capítulos de este documento se desarrolló un trabajo de investigación orientado al apoyo de las decisiones estratégicas. Para esto se identificaron y analizaron todos los posibles esquemas de adopción de IDN. Cada esquema fue investigado y ejemplificado mediante la experiencia de adopción bajo algún ccTLD.

También se realizó un trabajo de investigación técnico, donde se analizaron los principales estándares relacionados con IDN, y el estado del arte en las bibliotecas que implementan dichos estándares.

A continuación se aplicaron los conceptos investigados en el capítulo de diseño, planificando así los posibles esquemas de adopción: la inscripción regular o el *sunrise*. Se identificó que cada uno de estos esquemas puede ser combinado con el el Modelo de Variantes, generando posibles escenarios. En cada uno de los escenarios se asignaron las tareas necesarias para adaptar la plataforma de NIC Chile.

Además se realizó una implementación del estándar IDN y del Modelo de Variantes: la biblioteca NIC::IDN, que contiene las funcionalidades fundamentales para adaptar la plataforma de NIC Chile.

Finalmente, utilizando la biblioteca NIC::IDN se adaptó la plataforma de NIC Chile para operar

con caracteres extendidos. Para lograr esto, fue necesario implementar una serie de verificaciones automatizadas sobre la plataforma. Estas verificaciones permiten tener algún grado de confianza sobre la correctitud operacional del sistema de registro de dominios. Y en el futuro, si se continúan perfeccionando, pueden pasar a ser una parte indispensable en las verificaciones de la plataforma.

La incorporación de IDN en Chile es un tema pionero a nivel Latino Americano. Es importante destacar que la adopción de IDN es un tema complejo, presentando nuevos desafíos en el ámbito tecnológico. Sin embargo, la mayor dificultad consiste en compatibilizar los intereses de diversas áreas: técnica, legal, económica y gremial (CNNN<sup>1</sup>). A lo largo de esta memoria se ha intentado diseñar una solución que incorpore las inquietudes de todos estos ámbitos. Por esta razón, este documento representa una guía de apoyo y referencia, tanto para la toma de decisiones estratégico-políticas, como para las soluciones tecnológicas que deberán acompañar a estas decisiones.

## 7.2. Trabajo Futuro

El proceso de adopción de IDN bajo .CL se encuentra iniciado con el presente documento. Aún queda trabajo por realizar el cual se detalla a continuación:

1. Desarrollar interfaces para aplicar *sunrise*, en caso de que NIC Chile opte por este escenario.
2. Adaptar mecanismo de recepción de solicitudes para manejar ráfagas de formularios. Actualmente se encuentra en desarrollo por parte del equipo de Ingeniería de NIC Chile.
3. Robustecer el conjunto de *tests* automatizados de la plataforma NIC1.
4. Aplicar nuevas funcionalidades en desarrollo plataforma NIC3. Implementando, por ejemplo, el API de biblioteca NIC::IDN en su versión Java.
5. Capacitar a Atención al Cliente y Soporte Técnico de NIC Chile para responder consultas sobre IDN.

---

<sup>1</sup>Consejo Nacional de Nombres de Dominios y Números IP ([www.cnnn.cl](http://www.cnnn.cl))

# Bibliografía

- [1] DENIC “IDN FAQ”, [http://www.denic.de/en/faqs/idn\\_faqs/index.html](http://www.denic.de/en/faqs/idn_faqs/index.html)
- [2] DNS&BIND Albitz, Paul and Liu Cricket “DNS and BIND” 4th Edition, April 2001, O’Reilly & Associates.
- [3] ICANN IDN Guidelines June 2003, Version 1.0 <http://www.icann.org/general/idn-guidelines-20jun03.htm>
- [4] ICANN “IDN Workshop”, July 2004, Kuala Lumpur. <http://www.icann.org/meetings/kualalumpur/idn-workshop-08jul04.htm>
- [5] Kosmulski, Michal “Using Unicode in Linux”, November 2004. <http://software.newsforge.com/software/04/10/27/1631244.shtml?tid=130>
- [6] KRNIC IDN July 2004. <http://www.icann.org/presentations/ahngu-idn-kl-21jul04.pdf>
- [7] Simon Josefsson, “GNU IDN Library - Libidn” <http://www.gnu.org/software/libidn/>
- [8] MySQL Manual Unicode Support <http://dev.mysql.com/doc/mysql/en/Charset-Unicode.html>
- [9] NASK, “Internationalized Domain Names in Poland”, July 2004 ICANN IDN Workshop Kuala Lumpur. <http://www.icann.org/presentations/bartosiewicz-idn-kl-21jul04.pdf>
- [10] NASK “Allowed Character Sets”, February 2004. [http://www.dns.pl/IDN/allowed\\_character\\_sets.pdf](http://www.dns.pl/IDN/allowed_character_sets.pdf)
- [11] NICAT “IDN Introduction in Austria”, April 2004. [http://www.nic.at/en/docs/IDN\\_Statistiken\\_News.pdf](http://www.nic.at/en/docs/IDN_Statistiken_News.pdf)

- [12] RFC-3454 IETF “Preparation of Internationalized Strings ("stringprep")”, December 2002. <http://www.ietf.org/rfc/rfc3454.txt>
- [13] RFC-3490 IETF “Internationalizing Domain Names in Applications (IDNA)”, March 2003. <http://www.ietf.org/rfc/rfc3490.txt>
- [14] RFC-3491 IETF “Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)”, March 2003. <http://www.ietf.org/rfc/rfc3491.txt>
- [15] RFC-3492 IETF “Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA) “ , March 2003. <http://www.ietf.org/rfc/rfc3492.txt>
- [16] RFC-3912 IETF “Whois Protocol Especification”<http://www.ietf.org/rfc/rfc3912.txt>
- [17] Spolsky Joel , “Painless Software Schedules” March 29, 2000. <http://www.joelonsoftware.com/articles/fog0000000245.html>
- [18] TWNIC, “Experience on Whois Database” <http://www.icann.org/presentations/hsu-idn-kl-21jul04.pdf>
- [19] Wikipedia “Idiomas de Chile” [http://es.wikipedia.org/wiki/Idiomas\\_de\\_Chile](http://es.wikipedia.org/wiki/Idiomas_de_Chile)

# Apéndice A

## Guía ICANN para implementación de IDN.

La guía de ICANN intenta normalizar la metodología de los diversos registros TLD. La guía completa en su idioma original (inglés), se encuentra referenciada en la bibliografía de este documento[3]. A continuación se señalan los items más relevantes expresados en español.<sup>1</sup>

1. Los dominios de primer nivel (TLD) que implementen IDN lo harán en estricto rigor con las especificaciones técnicas descritas en los RFCs 3490, 3491 y 3492[13, 14, 15] (conocido colectivamente como estándar IDN).
2. Al implementar el estándar IDN, los registros TLD utilizarán un enfoque basado en la inclusión para identificar los caracteres Unicode permitidos. Es decir, aquellos caracteres que no son explícitamente permitidos deberán ser prohibidos.
3. Al implementar el estándar IDN, los Registros TLD (a) asociarán cada dominio IDN con una lengua o conjunto de lenguas, (b) emplearán procesos de registro y reglas de administración documentadas públicamente específicas para cada lengua, tales como reserva de todos los dominios con variantes de caracteres equivalentes en la lengua asociada a dicho dominio, y (c) cuando un Registro TLD encuentre que el registro y reglas de administración para un idioma dado se beneficie de una tabla de variante de caracteres, permitirá el registro en ese idioma solamente después de definir una tabla apropiada.
4. Los Registros TLD trabajarán colaborativamente con los actores relevantes e interesados en desarrollar políticas de registro específicas a una lengua (incluyendo, cuando el Registro determine apropiado, tablas de variantes de caracteres), con el objetivo de lograr un enfoque

---

<sup>1</sup>Esta no es una traducción fiel. Más bien, que intenta extraer los conceptos importantes de cada ítem.

consistente a la implementación de IDN, que beneficie a todos los usuarios del DNS a nivel mundial. Los Registros trabajarán colaborativamente los unos con los otros para abordar temas comunes, a través, por ejemplo de la creación de grupos de trabajo.

5. Al implementar el estándar de IDN, los Registros TLD deberán, inicialmente al menos, limitar los dominios a caracteres asociados con una lengua o conjunto de lenguas.
6. Los Registros de TLD (y Registradores) deberán proveer recursos de información y servicios en todos los lenguajes par los cuales se ofrecen registros de IDN.

# Apéndice B

## API Biblioteca NIC::IDN

Esta biblioteca se encuentra implementada en Perl como módulo. Además se encuentra definido el API de la biblioteca en Java para una futura implementación. Para cada método en un lenguaje, encontramos su equivalente en el otro.

### B.1. Validar Dominio o Variante

```
JAVA: public boolean isValidDomain(String domain, String version);  
JAVA: public boolean isValidDomain(String domain);  
PERL: sub isValidDomain domain version  
PERL: sub isValidDomain domain
```

**domain** Corresponde al FQDN de un dominio.

**version** Corresponde a la versión de la Tabla de Variantes que se desea utilizar para realizar la validación. En caso de no ser especificado se utilizará la tabla más reciente.

**retorno** Verdadero si el dominio se puede expresar en base a la Tabla de Variantes. Falso si es que no se puede.

## B.2. Representación Canónica

```
JAVA: public String getCanonico(String domain,String version);  
JAVA: public String getCanonico(String domain);  
PERL: sub getCanonico domain version  
PERL: sub getCanonico domain
```

**domain** Corresponde al FQDN para el cual se desea obtener su representación canónica.

**version** Corresponde a la versión de la Tabla de Variantes que se desea utilizar. En caso de ser omitido se utiliza la más reciente.

**retorno** Corresponde a la representación canónica de el dominio.

## B.3. Identificador de Dominio

```
JAVA: public String getIdentificador(String domain);  
PERL: sub getIdentificador domain
```

**domain** Corresponde al FQDN para el cual se desea obtener su identificador.

**retorno** Corresponde al identificador del dominio.

## B.4. Pertencia Misma Clase

```
JAVA: public boolean isSameClase(String domain1, String domain2);  
PERL: sub isSameClase domain1 domain2
```

**domain1** Corresponde al FQDN que se desea comparar.

**domain2** Corresponde al FQDN que se desea comparar.

**retorno** Verdadero si pertenecen a la misma Clase, falso en caso contrario.



## B.5. Equivalencia Canónica

```
JAVA: public boolean isEqualCanónico(String domain1, String domain2);
```

```
PERL: sub isEqualCanónico domain1 domain2
```

**domain1** Corresponde al FQDN que se desea comparar.

**domain2** Corresponde al FQDN que se desea comparar.

**retorno** Verdadero si pertenecen a la misma Clase, falso en caso contrario.

## B.6. Equivalencia

```
JAVA: public boolean isEqual(String domain1, String domain2);
```

```
PERL: sub isEqual domain1 domain2
```

**domain1** Corresponde al FQDN que se desea comparar.

**domain2** Corresponde al FQDN que se desea comparar.

**retorno** Verdadero si pertenecen a la misma Clase, falso en caso contrario.

## B.7. Obtener Clase

```
JAVA: public String[] getClase(String domain);
```

```
PERL: sub getClase domain
```

**domain** Corresponde al FQDN para el cual se desea obtener la Clase.

**retorno** Arreglo de Strings con toda la Clase a la que pertenece el dominio.

## B.8. Pertenencia a una Clase Forzada

```
JAVA: public boolean isClaseForzada(String domain);
```

```
PERL: sub isClaseForzada domain
```

**domain** Corresponde al FQDN para el cual se desea determinar si pertenece a una Clase Forzada.

**retorno** Arreglo de Strings con toda la Clase a la que pertenece el dominio.

## B.9. Determinar si un dominio es IDN

```
JAVA: public boolean isIDN(String domain);
```

```
PERL: sub isIDN domain
```

**domain** Corresponde al FQDN para el cual se desea determinar si corresponde a un dominio con caracteres tradicionales o extendidos.

**retorno** Verdadero si es un dominio con caracteres extendidos, falso si está compuesto solamente por LDH.

## B.10. Determinar si se puede codificar en ACE

```
JAVA: public boolean to_ascii_success(String domain);
```

```
PERL: sub to_ascii_success domain
```

**domain** Corresponde al FQDN para el cual se desea determinar si puede ser transformado en ACE.

**retorno** Verdadero si el algoritmo ToASCII no retorna error, falso en caso contrario.

# Apéndice C

## Listado de Archivos Modificados o Creados

### C.1. Host Virtual `www.nic.cl`

- `$niccl/nic-adm/mhonarc.pl`
- `$niccl/nic-adm/imprimir-certificado.pl`
- `$niccl/cgi-bin/ingresa-solicitud`
- `$niccl/cgi-bin/ultimos`
- `$niccl/cgi-bin/carro`
- `$niccl/templates/carro/servipag-unico.html`
- `$niccl/cgi-bin/boletas`
- `$niccl/cgi-bin/dom-CL`
- `~mleyton/cgi-bin/src/zone-grep/zone-grep.c`
- `~mleyton/cgi-bin/src/zone-grep/Makefile`
- `$niccl/cgi-bin/zone-grep`
- `~mleyton/cgi-bin/src/zone-grep-secundario/zone-grep.c`
- `~mleyton/cgi-bin/src/zone-grep-secundario/Makefile`
- `secundario@$niccl/cgi-bin/zone-grep`

- `~mleyton/cgi-bin/src/log-grep/log-grep-web.c`
- `$niccl/nic-adm/log-grep`
- `$niccl/htmls/faq/index.html`
- `$niccl/htmls/faq/idn.html`
- `$niccl/cgi-bin/idntool.pl`
- `$niccl/nic-adm/soportetools/tool.pl`
- `$niccl/nic-adm/soportetools/editform.pl`

## **C.2. Directorio CLCORREO**

- `$clcorreo/src/asigna`
- `$clcorreo/src/activa`
- `$clcorreo/src/whoisd`
- `$clcorreo/src/congela`
- `$clcorreo/src/protege`
- `$clcorreo/src/borra`
- `$clcorreo/src/reactiva`
- `$clcorreo/cli/bin/CLIO.activa`

## **C.3. Módulos de Perl**

- `$sitenic/Form.pm`
- `$sitenic/Auth.pm`
- `$sitenic/Util.pm`
- `$sitenic/IDN.pm`
- `$sitenic/tlds.pl`

- \$sitenic/MensajesChileno.pm
- \$sitenic/Conf.pm
- \$sitenic/Test.pm

# Apéndice D

## Documentos PDF

### D.1. Cupón de Pago

NIC Chile  
RUT: 60.910.000-1  
Agustinas 1357, piso 4  
Código Postal 6500587  
Santiago, Chile

Código de servicio Servipag: 78200

UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN  
**NIC Chile**  
Aviso de Cobranza

Talón NIC Chile

Dominio	pedigüehería
N° Proceso	20050304183631 4361303

Revise la información de su dominio ANTES DE PAGAR

Razón Social	MARIO HERNAN GUILLERMO LEYTON LUEJE
Atención de	MARIO HERNAN GUILLERMO LEYTON LUEJE

Fecha de Emisión	04/03/2005
Cancelar hasta	03/04/2005

Timbre de Caja  
Valor a pagar  
20.170.-

ESTE DOCUMENTO NO CONSTITUYE FACTURA, ÉSTA SE LE ENVIARÁ POR CORREO UNA VEZ CANCELADO

El pago se recibe exclusivamente en oficinas de SERVIPAG.

Si paga con cheque, extenderlo a la orden de "Universidad de Chile".

Si paga por Webpay NO PAGUE este cupón

Más información en  
<http://www.nic.cl>  
Consultas por email a [info@nic.cl](mailto:info@nic.cl)  
teléfono: (2) 940-7700  
fax: (2) 940-7701

Copia NIC Chile

Figura D.1: Ejemplo cupón de pago con IDN

## D.2. Factura Electrónica



### UNIVERSIDAD DE CHILE

Sucursal: NIC Chile (Codigo SII: 59545645)  
 Corporación Educacional y Servicios  
 Profesionales  
 Agustinas 1357, Cuarto Piso  
 Santiago, Santiago

R.U.T.: 60.910.000-1

FACTURA ELECTRONICA

N° 000072181

S.I.I. - SANTIAGO CENTRO

Santiago, 04 de Marzo de 2005

SEÑOR(ES): MARIO LEYTON

R.U.T.: 13.924.245-9

DIRECCION: Agustinas 1357 Piso 4, Santiago Tel:244 2724

COMUNA: Providencia

CIUDAD: Santiago

GIRO: Ingenieria

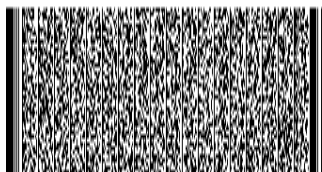
CONDICION VENTA: Efectivo (Contado)

CENTRO DE COSTO: 1966

ITEM: 6.1.01.03.01 (2152)

VENCIMIENTO: 04/11/2004

Cantidad	Detalle	P. Unitario	Total
1.0	dominio pedigüeñeria/20040922131414	16949.58	16950
Atencion de: Mario Leyton			
		Suma	16950
		IVA 19%	3220
		Total	20170



PAGADO  
 04 Mar 2005  
 NIC CHILE

Timbre Electrónico SII  
 Res.20 de 2003-Verifique documento:www.sii.cl

Figura D.2: Ejemplo Factura con IDN