Universidad de las Ciencias Informáticas.

Facultad 15

Implementación de un Motor de Codificación y Búsqueda Fonética para el Sistema de Gestión Integral de la Aduana.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor:

Carlos Manuel Romero González

Tutor:

Ing. Manuel Ramón Almaguer Ochoa

Consultante:

Msc. Julio Cesar Díaz Vera

Ciudad de la Habana, junio de 2010.
"Año 51 de la Revolución"

“Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el hombre nuevo que se vislumbra en el horizonte.”

“El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres.”

“No se trata de cuántos kilogramos de carne se come o de cuántas veces por año pueda ir alguien a pasearse por la playa, ni de cuántas bellezas que vienen del exterior puedan comprarse con los salarios actuales. Se trata, precisamente, de que el individuo se sienta más pleno, con mucha más riqueza interior y con mucha más responsabilidad.”

Ernesto Rafael Guevara de la Serna,
más conocido por Latinoamérica como:
Ernesto “Che” Guevara.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al proyecto de Sistemas Tributarios y Aduanas de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan uso pertinente del mismo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 20__.

_____	_____
Carlos Manuel Romero González	Ing. Manuel Ramón Almaguer Ochoa
Firma del Autor	Firma del Tutor

AGRADECIMIENTOS

Agradezco a todas aquellas personas que de una forma u otra siempre me han apoyado incondicionalmente, en especial a mi familia y mis amistades más íntimas.

Deseo agradecer además de una forma intrínseca a mi madre, Ana María González Salermo, por haberme sabido guiar en los momentos más difíciles de mi vida.

Gracias a ella hoy demuestro mi valía como hombre de este tiempo.

A Yisell Viamontes Lores
por compartir junto a mí tan agradables momentos.

A mis amigos: "the wild group"
Ricardo Pablo Ávila Alfaro,
Lázaro Manuel Gil Martínez,
Oscar Luis Garcell Martínez y su novia
Dairely De La Cruz San Juan,
y finalmente a nuestro consejero mayor
Jorge Alexis Del Castillo Palma.

A mis seres queridos que no están presentes por razones del destino.

A todos muchísimas gracias.

El Autor.

DEDICATORIA

Dedico este trabajo de diploma a mis seres queridos más cercanos como muestra fehaciente de que mi formación profesional se la debo en gran parte a ellos.

También lo dedico a la Universidad de las Ciencias Informáticas, institución donde me forjé como un hombre de futuro.

RESUMEN

La recuperación de información a partir de la base de datos del Sistema de Gestión Integral de la Aduana General de la República de Cuba, se está viendo eventualmente afectada debido a la determinación de todas las formas variantes de los nombres de las personas naturales almacenadas.

La dificultad fundamental de la identificación de nombres personales radica generalmente en la gran diversidad de estructuras de este tipo originadas por factores históricos y culturales. Además, no queda exenta la continua producción de errores de naturaleza ortográfica y tipográfica, durante los procesos de gestión de esta información, los cuales atentan contra la integridad gramatical de los datos.

Los métodos de codificación fonética constituyen uno de los procedimientos más utilizados para erradicar esta problemática. En este trabajo se realiza una revisión de los procesos que utilizan los sistemas Soundex, Daitch-Mokotoff Soundex, NYSIIS, Phonix y Metaphone, para la asignación de claves fonéticas, permitiendo reducir a una forma común aquellos nombres personales que son similares en cuanto a pronunciación.

La objetividad fundamental enmarcada dentro del estudio de estos algoritmos de equiparación aproximada de nombres correspondientes a personas naturales, está dirigida al diseño e implementación de un motor de codificación y búsqueda fonética para el sistema utilizado por la aduana, basado en la normalización canónica de los sonidos vocálicos y consonánticos del idioma español, contribuyendo de esta forma a eliminar las inconsistencias originadas por ambigüedades de esta naturaleza.

TABLA DE CONTENIDO

TABLAS Y FIGURAS

Introducción

La Aduana General de la República de Cuba (AGR) está implicada actualmente en un proceso de informatización que abarca la mayoría de sus sectores. Las aplicaciones informáticas que están siendo utilizadas hasta el momento forman parte, en su conjunto, del Sistema de Gestión Integral de la Aduana (GINA), que está caracterizado por manejar Información Clasificada a diferentes niveles, causa por la cual puede variar mucho la gestión de dicha información de un nivel o sector a otro, aunque estos estén estrechamente relacionados y unidos por una misma base de datos (BD). Estas aplicaciones también poseen como característica principal, que tributan en gran medida al manejo de información de personas naturales almacenadas en su BD centralizada. Atributos personales vitales como nombres, apellidos y nacionalidad, están expuestos a diferentes tipos de errores ya sean ortográficos o tipográficos, que dan lugar a omisiones, inserciones o sustituciones en los caracteres que componen dichas cadenas, razón por la cual se afecta en gran medida la recuperación o búsqueda de información que se realiza sobre el dominio correspondiente.

Con el remarcable progreso de las TIC[1] se ha demostrado que los métodos de codificación fonética tienen gran utilidad en varios campos, especialmente en la gestión, permitiendo de esta forma realizar búsquedas en un amplio dominio que comprende las diferentes variantes de los nombres, apellidos o localidades, basándose principalmente en la pronunciación de la palabra en lugar su grafía. A consecuencia de esta propiedad surgen un conjunto de aplicaciones relacionadas específicamente con la necesidad de identificar las variantes de los nombres personales en los distintos Sistemas de Recuperación de Información (SRI), tomando como ejemplo en este caso, las bibliotecas digitales, las bases de datos de pacientes de un hospital, los sistemas de reservas aéreas y los sistemas de censo poblacional [1].

[1] Tecnologías de la Informática y las Telecomunicaciones.

Las correlaciones entre cómo suena una palabra y cómo se escribe no son triviales, añadiendo además que cada idioma tiene sus propias reglas de pronunciación. Los métodos de codificación fonética han sido ajustados a diferentes algoritmos, los cuales utilizan la teoría de conjuntos para dividir, dado un dominio, todos los posibles nombres y apellidos en diversos subconjuntos fonéticamente cercanos, utilizando para esto, las reglas de pronunciación de un idioma previamente definido. Este principio forma parte de la esencia y el aporte investigativo de este trabajo de diploma, en el cual se define un alcance netamente práctico y palpable orientado a la implementación de un sistema de codificación y búsqueda fonética aplicado directamente al Sistema GINA.

Situación problemática

La disponibilidad e integridad operativa de la información de las personas naturales almacenadas en la BD de la aduana mediante sus procesos de gestión y control, se ve afectada directamente por los errores antes mencionados a pesar de conocer las causas de su origen. La necesidad de garantizar y satisfacer las exigencias operacionales de dicha entidad para llevar a cabo estos procesos, requiere en estos momentos de un conjunto de aplicaciones que gestionan y controlan de forma automática gran parte de esta información. El procesamiento automatizado de información se rige por un conjunto de reglas y validaciones que ya fueron definidas en su momento mediante un análisis exhaustivo del proceso negocio en cuestión, sin embargo no es posible garantizar la integridad gramatical de datos valiosos como nombres y apellidos, lo cual genera algunas inconsistencias que afectan a corto plazo el dominio utilizado por las aplicaciones del Sistema GINA.

La AGR en calidad de su legislatura comienza a recibir a partir del año 2007 la Información Adelantada de Pasajeros y Tripulantes (API), la cual se traduce en un mensaje que envían las aerolíneas, luego del despegue del avión y que trae consigo los datos de los pasajeros que están en el vuelo [2]. Estos datos son gestionados partiendo del procesamiento automático de dichos mensajes, recibidos a través de la red de la Sociedad Internacional de Comunicaciones Aeronáuticas (SITA), correos electrónicos o conformados manualmente mediante el acceso a la aplicación pertinente desde internet.

La estructura de estos mensajes se torna muy compleja y por tanto costosa de procesar en cuanto a recursos de cómputo, esto dificulta el uso de otras técnicas de validación gramaticales tales como el uso

de diccionarios electrónicos con el fin de corregir las faltas ortográficas que puedan ser almacenadas junto a las cadenas de datos procesadas, además se hace imposible analizar mediante la corrección ortográfica todas las variantes de nombres propios de personas existentes en el mundo, esto sin contar otros errores de naturaleza sintáctica que puedan ser pasados por alto como el uso incorrecto de mayúsculas y la alteración del significado debido a la distinta distribución de los componentes del nombre propio [1].

Esta información procesada a través del API es muy valiosa para los analistas que operan el sistema de control en materia de enfrentamiento a los lícitos aduaneros que se realiza por la especialidad de Lucha Contra el Fraude (LCF). Un sistema que proporciona múltiples herramientas de control que permiten identificar los hechos, indicios e ilícitos en que incurren personas naturales consideradas de interés aduanal, procedentes del tráfico mercantil, viajero o postal, permitiendo detectar posibles violaciones de las leyes migratorias vigentes en el país y donde es de vital importancia trabajar con datos específicos muy relevantes como es el caso de los nombres propios de las personas. Por consiguiente, la precisión de la identificación de estas personas depende la calidad de la información con que cuenta la aplicación y de su nivel de confiabilidad [2].

Esta situación también se extiende al área de los recursos humanos de la AGR, y aunque se manifiesta en menor grado, cabe la posibilidad de que las validaciones existentes hasta el momento no logren realizar una equiparación exacta entre un nuevo individuo procesado con el objetivo de formar parte de un área de trabajo determina y las posibles variantes de los nombres personales existentes en la misma, lo cual provocaría que en algunas ocasiones se dificulte comprobar si ese futuro trabajador de la entidad ha sido procesado anteriormente por la aduana cubana o presenta otros antecedentes relacionados con viajes al exterior del país u otra causa donde los identificadores o unificadores como es el caso del número de carné de identidad no desempeñen un papel fundamental.

En múltiples módulos de estas aplicaciones de gestión aduanera se hace necesario hacer búsquedas utilizando criterios formados por cadenas no muy largas que representan nombres o apellidos de personas, en las cuales se establece una correspondencia entre los términos de la consulta y la forma de dichos nombres pre-almacenada. La dificultad radica cuando no se conoce exactamente como se escriben estos nombres o apellidos [1].

Como se ha alegado con anterioridad, la garantía de que la información referente a las personas se almacene correctamente en la BD del Sistema GINA se ve afectada de manera importante, atendiendo a la dinámica actual de gestión existente en la AGR y a las diferentes aplicaciones que realizan operaciones de inserción, actualización y consulta sobre ese dominio específico de la entidad.

Problema

El Sistema GINA no garantiza actualmente la unicidad e integridad de los datos de las personas almacenadas en su base de datos. Tributando de esa forma a diversas ambigüedades en informaciones vitales como nombres y apellidos.

Objeto de estudio

El objeto de estudio comprende los algoritmos de codificación fonética basados en la clasificación y unificación de sonidos vocálicos y consonánticos del abecedario [3].

Campo de acción

Los algoritmos de codificación fonética del idioma español, aplicados directamente a las variantes de los nombres y apellidos de las personales naturales almacenadas en la base de datos de la AGR.

Objetivo general

Implementar un motor de codificación y búsqueda fonética para el Sistema de Gestión Integral de la Aduana que utilice las técnicas de equiparación basadas en las reglas de clasificación y unificación de sonidos vocálicos y consonánticos del idioma español.

Objetivos específicos

- Identificar y transcribir a un lenguaje de programación previamente definido, todas las reglas de normalización basadas en estudios fonológicos del idioma español.

- Identificar y editar en dicho lenguaje, la organización de los diferentes grupos de grafemas del alfabeto castellano que presentan similitud fonética, así como el fonema representativo asignado a cada uno de estos grupos.
- Determinar cuáles de los algoritmos existentes se ajusta mejor para la asignación de claves fonéticas utilizando las reglas de normalización fonológicas identificadas.
- Establecer una línea de diseño donde se defina el funcionamiento del algoritmo y la utilización de las reglas de forma independiente, de tal manera que si se crean nuevas reglas en estudios posteriores, estas puedan ser agregadas sin dificultad.
- Implementar la solución propuesta utilizando las tecnologías, métodos y herramientas definidos durante el diseño.

Tareas de investigación

- Fundamentación del estudio sobre los diferentes sistemas de codificación fonética, enfocado en las aplicaciones y limitaciones existentes.
- Estudio sobre otras técnicas gramaticales aplicables a la problemática analizada para identificar las principales ventajas de la codificación fonética sobre estas primeras.
- Obtención de los fundamentos teóricos del idioma español sobre los cuales se basan las reglas de normalización de nombres personales y la organización de los grupos de similitud fonética de las consonantes del alfabeto.
- Comprensión del principio de funcionamiento de los diferentes algoritmos de codificación fonética y sus relaciones lenguaje-dependientes con las reglas definidas por el idioma para el cual fueron creados.
- Identificación y descripción de las características del sistema sobre el cual se aplicará la solución propuesta.
- Descripción de los diferentes patrones y herramientas de diseño de Software aplicables a la estructura general del la solución a implementar.
- Estudio y utilización de técnicas de optimización algorítmica durante el desarrollo de la solución planteada.
- Diseño y aplicación de casos de prueba con la finalidad de comprobar el funcionamiento del motor una vez concluido.

Cronograma de ejecución

ID		Task Name	Duration	Start	Finish
1	✓	Fundamentación del estudio sobre los diferentes sistemas de codificación fonética, enfocado en las aplicaciones y limitaciones existentes.	5 days	Thu 21/01/10	Wed 27/01/10
2	✓	Estudio sobre otras técnicas gramaticales aplicables a la problemática analizada para identificar las principales ventajas de la codificación fonética sobre estas primeras.	1 day	Sat 30/01/10	Mon 01/02/10
3	✓	Obtención de los fundamentos teóricos del idioma español sobre los cuales se basan las reglas de normalización de nombres personales y la organización de los grupos de similitud fonética de las consonantes del alfabeto.	7 days	Wed 03/02/10	Thu 11/02/10
4	✓	Comprensión del principio de funcionamiento de los diferentes algoritmos de codificación fonética y sus relaciones lenguaje-dependientes con las reglas definidas por el idioma para el cual fueron creados.	4 days	Wed 10/02/10	Mon 15/02/10
5	✓	Identificación y descripción de las características del sistema sobre el cual se aplicará la solución propuesta.	2 days	Mon 15/02/10	Tue 16/02/10
6	✓	Descripción de los diferentes patrones y herramientas de diseño de software aplicables a la estructura general del la solución a implementar.	10 days	Thu 18/02/10	Wed 03/03/10
7	✓	Estudio y utilización de técnicas de optimización algorítmica durante el desarrollo de la solución planteada.	11 days	Tue 02/03/10	Tue 16/03/10
8	✓	Diseño y aplicación de casos de prueba con la finalidad de comprobar el funcionamiento del motor una vez concluido.	40 days	Mon 12/04/10	Sat 05/06/10

Estructuración del contenido

El presente documento está compuesto por tres capítulos que dan respuesta a las tareas de investigación anteriormente descritas. Cada uno de ellos contiene los elementos necesarios que conforman en su conjunto la solución final, partiendo el estudio del problema planteado.

Durante el transcurso del capítulo I se complementan los aspectos teóricos más relevantes de los sistemas de búsqueda basados en la fonetización de las cadenas de caracteres, así como el estudio de sus principales características y limitaciones, partiendo además del principio de funcionamiento de los algoritmos de normalización y codificación canónica basados en los grupos de similitud fonética definidos para el idioma español. De esta forma se persigue como objetivo principal el planteamiento de una propuesta de solución acorde con el problema planteado.

Esta propuesta de solución se describe en el capítulo II, y está enmarca fundamentalmente en la arquitectura definida para el actual Sistema de Gestión Integral de la Aduana. Serán detallados también durante el transcurso del mismo, los aspectos fundamentales del diseño, así como las tecnologías, métodos y herramientas utilizados para lograr la materialización de dicha solución.

Durante el transcurso del tercer (III) último capítulo de este trabajo, se abordan temas correspondientes al desarrollo de la solución, tales como las técnicas de optimización algorítmica para disminuir sus demandas computacionales, así como la organización e interacción de los componentes utilizados durante su construcción. También se notificarán los resultados de un conjunto de pruebas unitarias realizadas durante las diferentes fases de construcción, comprendidas entre la transcripción de las reglas de normalización canónica y el desarrollo del algoritmo de asignación de códigos fonéticos, además de una prueba de funcionamiento del motor durante los procesos de inserción, actualización y búsqueda, tanto en entornos controlados como de despliegue piloto.

1. CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

Durante el transcurso de este capítulo se complementan los aspectos teóricos más relevantes de los sistemas de búsqueda basados en la fonetización de las cadenas de caracteres, así como el estudio de sus principales características y limitaciones, partiendo además del principio de funcionamiento de los algoritmos de normalización y codificación canónica basados en los grupos de similitud fonética definidos para el idioma español. De igual forma serán analizadas también en esta sección la existencia de otras medidas de equiparación aproximada de cadenas fundamentadas mediante el uso de sistemas de corrección ortográfica y las principales ventajas de la codificación fonética sobre esta práctica.

Estas acciones complementan objetivamente la identificación de las principales reglas de normalización fonética del idioma castellano, basadas en los estudios fonológicos de los sonidos que conforman los diferentes grupos de consonantes que presentan similitud entre sus fonemas representativos, permitiendo a su vez conformar una propuesta de solución acorde con el problema planteado, a partir de la selección del algoritmo idóneo que se encargará posteriormente de brindar soporte a la misma.

1.1. Estado del arte.

La existencia de los sistemas de codificación y búsqueda fonética obedece a la necesidad de recuperar información que tiene una semejanza sonora; y cuya representación a través de la palabra escrita pueda diferir de su pronunciación [4].

1.1.1. Surgimiento de la búsqueda fonética. Sistemas vigentes.

Desde que se efectuaron los censos de 1890-1920 en Estados Unidos, se hizo necesario posteriormente realizar un análisis retrospectivo de esta información con el objetivo de lograr una mayor confiabilidad en cuanto a su forma original de registro, este nuevo principio de recuperación requería de una aproximación que fuese lo más exacta posible al criterio de partida para emprender búsquedas comprendidas dentro del dominio. Por este motivo fue utilizado en 1930 la primera variante del algoritmo Soundex, identificado como American Soundex, con la finalidad de equiparar la información de los nombres personales registrados durante el censo y agrupar las búsquedas de forma diferenciada partiendo de sus similitudes fonéticas, permitiendo así tener un grado de exactitud muy elevado durante la recuperación.

La utilización de estos algoritmos ofrecen soporte actualmente para un conjunto de interfaces de búsqueda avanzada, implementadas a partir de las necesidades específicas de diferentes sistemas entre los que se incluyen aplicaciones de búsqueda en bibliotecas virtuales como es el caso de AquaBrowser Library e Hibernate Search. La codificación fonética ha sido extendida también a otras especialidades, por lo que se emplea a menudo en algunos traductores potentes como el desarrollado por Google. Además, proporcionan las herramientas de corrección ortográfica de buscadores reconocidos en Internet como Yahoo y SCIRUS Scientific Information, para corregir los criterios de búsqueda cuando son ingresados erróneamente.

1.1.1.1. Sistemas de marcas y signos distintivos.

Las interfaces de búsqueda de estos sistemas no presentan variaciones significativas con respecto a otros sistemas conocidos, implementan de manera estandarizada un motor de búsqueda general, el cual realiza exploraciones dentro de un determinado dominio, a través de palabras exactas, prefijos o sufijos de palabras y partes de una palabra. Sin embargo la clasificación sonora de las palabras juega un papel importante dentro de estos sistemas, una nueva marca o firma empresarial no debe pronunciarse igual o semejante a otras existentes o reconocidas mundialmente. Por lo tanto la fortaleza de estos sistemas depende fundamentalmente de los motores de búsqueda fonética desarrollados para este fin.

La tendencia actual de estos sistemas han cobrado un gran auge en el mundo de las aplicaciones informáticas, tal es el caso del Instituto Nacional Defensor de la Competencia y de la Protección de la Propiedad Intelectual (INDECOPI), el cual ha desarrollado un sistema de búsqueda fonética muy eficaz. Esta institución tiene el objetivo de proteger todas las formas de propiedad intelectual: desde los signos distintivos y los derechos de autor, hasta las patentes y la biotecnología.

El sistema utilizado por el gobierno australiano, conocido por el nombre de Australian Trade Mark On-line Search System (ATMOSS)[2], incorpora también esta variedad de casos de búsqueda y añade además la funcionalidad de diferenciación de imágenes. Aunque no las integra en una sola interfaz, porque la

[2] http://pericles.ipaustralia.gov.au/atmoss/falcon.application_start.

búsqueda general utilizaba un esquema avanzado que le permite manejar combinaciones complejas de palabras.

Webmarks, constituye otro sistema desarrollado con la finalidad de ser un buscador de marcas y signos distintivos para un estudio de abogados que mantiene información de sus clientes, los cuales requieren de un procedimiento de detección de nombres de empresas de pronunciación similar. En este caso, la herramienta fonética implementada es de gran utilidad para proveer un acercamiento a los probables infractores que utilizan la semejanza fonética como competencia desleal, para que el consumidor pueda confundirse con el producto verdadero.

Estos sistemas permiten analizar un espectro mayor de alcance, a diferencia de otros sistemas de corrección ortográfica, pues las palabras recogidas dentro de sus resultados de búsqueda, no proceden de una falta de escritura, sino de la equiparación sonora de los términos buscados contra otros ya existentes.

Muchos de estos sistemas utilizan las funciones nativas del algoritmo Soundex desarrollado en lenguaje PHP[3], y dentro del proceso se realiza la búsqueda fonética en cada una de las palabras concernientes a un espacio o dominio dado. En evaluaciones realizadas recientemente, el registro de marcas conocidas como Macdonald y Sony Erickson, entre otras, fue analizado utilizando términos muy similares, aunque con diferente grafía. En la prueba se plantearon los niveles de exactitud en cada uno de los términos relacionados [4].

1.1.1.2. Sistemas de gestión.

Las aplicaciones de gestión utilizan los métodos de codificación fonética para simplificar la búsqueda en las bases de datos cuando sólo se conoce la pronunciación de un nombre propio pero no su trascripción exacta. En general, estos sistemas parten de la suposición de que los nombres que comparten la misma clave se podrían considerar similares y se han utilizado principalmente en aplicaciones que involucran la

[3] Acrónico recursivo que significa **PHP Hypertext Preprocessor**.

identificación de nombres personales, tales como búsquedas en bases de datos bibliográficas, bases de datos de pacientes en un hospital, sistemas de reservas aéreas y de censo poblacional [1].

Recientemente Microsoft ha desarrollado múltiples aplicaciones empresariales entre las que se encuentra SharePoint Server 2010, un Software destinado fundamentalmente a mejorar la búsqueda empresarial, incorpora diversas capacidades de búsqueda incluida la fonética. Está diseñado con el objetivo de que los administradores de búsqueda puedan configurar una infraestructura de búsqueda segura y óptima que permita a los usuarios finales encontrar información en la empresa de forma rápida y eficaz [5].

Dentro de otros ámbitos de Software de gestión empresarial sobresalen algunos como Facturation y Stock [6] y CAS GenesisWorld: Vista General de Funciones [7], realizados por otras compañías con altos indicadores de calidad en ingeniería de Software para empresas conocidas con los nombres de Netmasters y Genesis World respectivamente. Ambos, equipados con un avanzado sistema de búsqueda global y búsqueda fonética.

1.1.2. Algoritmos de codificación fonética conocidos. Principales características.

La Tabla 1.1 enlista los algoritmos fonéticos más comunes así como la fecha en que fueron propuestos.

ALGORITMOS FONÉTICOS

Método	Año
Soundex	1918, 1930
Daitch-Mokotoff Soundex	1985
NYSIIS	1970
Phonix	1988, 1990
Metaphone	1990
Double Metaphone	2000

Tabla 1.1: Cronología de los principales algoritmos fonéticos conocidos.

"Expresado en teoría de conjuntos, los algoritmos fonéticos dividen el conjunto de todos los posibles apellidos en diversos subconjuntos, agrupando en cada subconjunto apellidos fonéticamente cercanos" [8].

Soundex por ejemplo crea subconjuntos disjuntos: es decir, un mismo apellido no puede pertenecer simultáneamente a dos o más de estos subconjuntos (con lo que se crea una partición del conjunto de todos los apellidos). Otros algoritmos, como el Daitch-Mokotoff, crean subconjuntos no disjuntos, lo cual significa que un mismo apellido puede estar simultáneamente en varios de estos conjuntos. En los siguientes epígrafes son detalladas las características y limitaciones fundamentales de varios de estos métodos [8].

1.1.2.1. Algoritmo Soundex.

"Desarrollado en 1918 por Robert Russell y Margaret Odell. Inicialmente el método fue utilizado para manipular el censo y datos de inmigración. Se volvió popular con el volumen III de "The Art of Computing". Actualmente también es parte de los algoritmos de búsqueda, se emplea en programas de manejo de bases de datos y programas para comprobar ortografía, entre otros" [8].

"El método usado por Soundex está basado en la clasificación fonética de los sonidos del habla humana, los cuales se dividen en 6 clases: bilabial, labiodental, dental, alveolar, velar y glotal. Esta categorización depende de donde se colocan los labios y la lengua para generar un sonido" [8].

Este algoritmo transforma los apellidos ingleses en un código de cuatro caracteres. El primer carácter es una letra mayúscula y los tres restantes son dígitos. A continuación se presentan las características del algoritmo.

- ✓ **Características.**

  - La entrada del algoritmo Soundex estará dada por una cadena que representa el apellido de una persona, y la salida será la primera letra seguida por varios números.
  - La longitud de la salida no excederá los cuatro caracteres.
  - Los grafemas Y, W, H y las vocales, son ignoradas a menos que el apellido empiece por una de ellas, en cuyo caso, solo esa primera letra será tenida en cuenta.

- Elimina cualquier repetición consecutiva de caracteres.
- Las consonantes son sustituidas por códigos numéricos del 1 al 6, según los grupos de similitud sonora en el que se incluyan.

✓ **Tabla de codificación fonética.**

ALGORITMO SOUNDEX

Código	Caracteres
0	a e h i o u w y
1	b f p v
2	c g j k q s x z
3	d t
4	L
5	m n
6	R

Tabla 1.2: Grupos de similitud fonética de grafemas utilizados por Soundex.

	Apellidos	Variantes	Códigos
Match	Appelt	Apelt	(A143, A143)
	Hobbs	Hubbs	(H120, H120)
Mismatch	Appelt	Appell	(A143, A140)
	Hobbs	Hobds	(H120, H130)

Tabla 1.3: Resultados de la codificación realizada por el algoritmo Soundex.

✓ **Limitaciones presentadas.**

"Usando Soundex en los nombres Robert y Rupert se obtiene el mismo código (R163). Sin embargo, para nombres como Catherine y Katherine obtenemos códigos diferentes (C365) para Catherine y (K365) para Katherine" [8].

Otra de las inconveniencias presentadas por el algoritmo se registró cuando el codificador original de Soundex, no cifró la letra H al no considerar esta consonante como separador entre los grafemas adyacentes con el mismo código S y C, presente en algunos apellidos ingleses como "Ashcraft" [1].

✓ **Versiones mejoradas.**

Existen algunas versiones mejoradas de Soundex como Extended Soundex Algorithm [1] y Enhanced SoundEx la cual permite elegir el tamaño de la codificación final de cuatro a diez códigos, además de

emplear nuevas reglas para tratar de forma apropiada ciertas combinaciones de letras [8]. En la siguiente sección se explica otra versión de Soundex que genera una codificación diferente a la original.

1.1.2.2. Algoritmo Daitch-Mokotoff Soundex.

"Este algoritmo fue desarrollado en 1985 por el genealogista Gary Mokotoff y publicado en el primer número de "Avoyaynu" (el diario de la genealogía judía) en un artículo titulado "The Jewish Soundex: a revised format". Posteriormente Randy Daitch expandió las reglas del algoritmo creado por Mokotoff. Tanto Daitch como Mokotoff pertenecían a la Sociedad Genealógica Judía. Al final, este algoritmo es una mejora del algoritmo Soundex creado por Russell y Odell" [8].

El nuevo sistema surgió por la necesidad de indexar los nombres de unas 28,000 personas que vivían en Palestina entre 1921 y 1948. "Al realizar el censo, se observó que una gran cantidad de apellidos de gente judía eran apellidos alemanes y eslavos, los cuales tenían variaciones ortográficas y era necesario homogeneizarlos. Al utilizar el sistema Soundex que tenía el gobierno de Estados Unidos basado en el sistema de Russell y Odell, se observó que nombres diferentes con igual pronunciación no eran agrupados por un mismo código. Fue cuando surgió el método inventado por G. Mokotoff" [8].

- ✓ **Características:**

  - Las letras tienen diferentes valores si están al principio de la palabra o en el centro y varían si preceden a una vocal o no.
  - El código resultante posee una longitud de seis dígitos, en caso de que la cadena codificada no cubra tal longitud, serán completadas las cifras restantes con el valor cero.
  - La letra inicial es codificada como cualquier otra letra en el nombre. Si es una vocal, tiene el valor de cero.
  - Se agregaron varias reglas a tabla de codificación[4] utilizada por algoritmo para transcribir las secuencias de caracteres como dígitos individuales.

[4] La tabla de codificación se muestra en el Anexo 1.1.

- Puede generar varios códigos para un mismo nombre.

✓ **Principales aportes realizados.**

Las nuevas reglas del método Daitch-Mokotoff Soundex son independientes de consideraciones geográficas o étnicas. Y se ha convertido en un estándar utilizado por la Organización Genealógica Judía y la Sociedad de Ayuda a inmigrantes hebreos. La Tabla 1.4 muestra algunos ejemplos de la conversión de apellidos con ambos métodos.

CODIFICACIÓN FONÉTICA

Apellido	Soundex	D-M Soundex
Peters	P362	739400, 734000
Peterson	P362	739460, 734600
Moskowitz	M232	645740
Moskovitz	M213	645740
Auerbach	A612	097500, 097400
Uhrbach	U612	097500, 097400
Jackson	J250	154600, 454600, 145460, 445460

Tabla 1.4: Ejemplos de apellidos codificados con los algoritmos Soundex y Daitch-Mokotoff Soundex.

✓ **Limitaciones presentadas.**

Aunque constituye un estándar en este sentido, el Sistema Daitch-Mokotoff Soundex se caracteriza además por ser un algoritmo extremadamente complejo y por tanto, difícil de adaptar a otros sistemas que pretendan utilizar este tipo de codificación enmarcada en otros idiomas como el español y el portugués.

1.1.2.3. Algoritmo NYSIIS.

Código fonético propuesto por Taft en 1970, y desarrollado posteriormente por la New York State Division of Criminal Justice, denominado New York State Identification and Intelligence Systems (NYSIIS). Se

basaba en la normalización de los nombres personales ingleses a un código de 6 letras [4]. Las reglas utilizadas por el algoritmo NYSIIS para la codificación fonética se muestran a continuación:

- ✓ **Características.**

  - El primer carácter de la clave fonética corresponde al primer carácter del nombre.
  - Traduce los primeros y los últimos caracteres del nombre, respondiendo a las reglas de normalización y codificación fonética mostradas en la Tabla 1.5.
  - Para otros casos en que los últimos caracteres sean S ó A, entonces estos serán eliminados de la cadena.

- ✓ **Tabla de codificación fonética.**

ALGORITMO NYSIIS

Caracteres de Inicio		Caracteres Finales	
Código	Nombre	Código	Nombre
MCC	Mac	Y	ee, ie, ay
FF	Ph	D	dt, rt, nt, rd, nd
NN	Kn		
C	K		
SSS	Sch		

Tabla 1.5: Reglas de normalización y codificación de cadenas definidas para NYSIIS.

	Apellidos	Variantes	Códigos
Match	Appelt	Apelt	(APALT, APALT)
	Hobbs	Hubbs	(HAB, HAB)
Mismatch	Appelt	Appell	(APALT, APAL)
	Hobbs	Hobds	(HAB, HABD)

Tabla 1.6: Resultados de la codificación realizada por el algoritmo NYSIIS.

- ✓ **Limitaciones presentadas.**

El algoritmo NYSIIS solo trata las variantes de los nombres producidas por errores fonéticos, esto se debe a que el tipo de normalización de cadenas utilizada no se ajusta a las reglas de pronunciación de la lengua inglesa. Motivo por el cual en 1998 The New York State Division of Criminal Justice sustituye el sistema NYSIIS por el producto NameSearch, por medio del cual no solo se identifican las variantes fonéticas sino las producidas por errores de transcripción, formas abreviadas, o variantes originadas por la distinta ordenación de las secuencias de los componentes que forman los nombres personales [1].

1.1.2.4. Algoritmo Phonix.

El auténtico problema de los sistemas anteriores es que no son capaces de establecer algún tipo de ordenación entre las cadenas similares. Este problema se resuelve con una variante de Soundex llamada Phonix [1].

- ✓ **Características.**

Al igual que su predecesor (Soundex), el método de codificación Phonix se basa en la sustitución de todos los caracteres excepto el primero por valores numéricos. La novedad que introduce este algoritmo es que realiza previamente unas 163 transformaciones de grupos de letras que normalizan las cadenas, haciendo de este un sistema de codificación fonética de gran complejidad.

- ✓ **Tabla de codificación fonética.**

ALGORITMO PHONIX

Código	Caracteres
0	a e h i o u w y
1	b p
2	c g j k q
3	d t
4	l
5	m n
6	r
7	f v
8	s x z

Tabla 1.7: Grupos de similitud fonética de grafemas utilizados por Phonix.

- ✓ **Principales aportes realizados.**

El aporte más importante realizado por este sistema de codificación es que computa los sonidos finales de las palabras, siendo capaz de establecer tres rangos de similitud constituidos por cadenas que concuerdan en los sonidos finales, así como en los prefijos y sufijos de estos sonidos.

✓ **Limitaciones presentadas.**

Al igual que el sistema de Daitch y Mokotoff, el algoritmo fonético Phonix, dado su alto grado de complejidad, es difícilmente adaptable a otros sistemas que requieran para su funcionamiento una codificación basada en un idioma diferente. Una adaptación de tal magnitud daría como resultado un método completamente diferente al original, sin contar el costo en tiempo requerido para esta tarea.

1.1.2.5. Algoritmo Metaphone.

Se trata de un sistema de codificación especialmente diseñado para el inglés americano [1]. Fue desarrollado por Lawrence Phillips y dado a conocer en diciembre de 1990 como parte de una clase de algoritmo llamado Phonetic Matching o Phonetic Encoding [4].

✓ **Características.**

- El algoritmo de Metaphone elimina las vocales, aunque éstas permanecen si son la primera letra de una palabra.
- Retiene solamente las consonantes, que se reducen a 16 grafemas sin incluir los dígitos (aunque existen excepciones como el dígito '0' para representar el sonido /th/).
- Las apariciones de algunas consonantes consecutivas son sustituidas por un fonema unificador, por ejemplo el sonido formado por la combinación PH se sustituye por el fonema /f/.
- Las reglas[5] de codificación utilizadas por este algoritmo representarían aproximadamente cómo se pronuncia un nombre a partir de las definiciones de pronunciación de la lengua inglesa.
- Elimina las consecuencias de caracteres iguales de la cadena procesada.

El código liberado por este algoritmo está constituido fundamentalmente por las consonantes del alfabeto, aunque mantiene invariable la vocal inicial de la palabra procesada. El resultado de la aplicación del algoritmo Metaphone se muestra en la Tabla 1.8.

[5] Estas reglas están recogidas en el Anexo 1.2.

ALGORITMO METAPHONE

	Apellidos	Variantes	Códigos
Match	Appelt	Apelt	(APLT, APLT)
	Hobbs	Hubbs	(HBS, HBS)
Mismatch	Appelt	Appell	(APLT, APL)
	Hobbs	Hobds	(HBS, HBTS)

Tabla 1.8: Resultados de la codificación realizada por el algoritmo Metaphone.

- ✓ **Principales aportes realizados.**

Al normalizar solamente las consonantes del abecedario se reduce significativamente el tiempo de ejecución de Metaphone. Aspecto muy importante a tener en cuenta para valorar el uso de este algoritmo de codificación en sistemas que realizan búsquedas fonéticas en dominios avanzados que almacenan grandes volúmenes de información referente a personas naturales. Es necesario subrayar además que el lenguaje de programación PHP incorpora el método Metaphone basado en las reglas de normalización de la lengua inglesa a partir de su versión 4.0.0 como parte de su repositorio de funciones.

- ✓ **Versiones mejoradas.**

Luego de un profundo análisis realizado por Lawrence Phillips, tras recibir varias notificaciones de algunos errores de codificación producidos por el algoritmo Metaphone, publica en la revista “Computer Language”, en el año 2000, una versión con sustanciales mejoras entre las cuales introdujo un conjunto de reglas de codificación fonética haciendo más extensible la equiparación sonora realizada hasta ese momento por la versión original. Las nuevas reglas, fundamentadas en la base de un profundo estudio fonológico del idioma inglés, son capaces actualmente de tratar las consonantes que se pronuncian de manera diferente en variantes de palabras similares en cuanto a escritura, pues el idioma en cuestión tiene una tendencia a acumular un gran número de nombres personales a partir de fuentes pertenecientes

a otros idiomas como: francés, latín y griego. Por tal razón este nuevo algoritmo devuelve en ocasiones hasta dos llaves para las palabras que plausiblemente se pueden pronunciar en más de una forma, causa por la cual adopta el nombre de “Double Metaphone” [9].

1.2. Fundamentos teóricos.

Los buscadores fonéticos utilizan algoritmos de codificación para realizar transformaciones en cadenas no muy largas basados en la similitud fonética de los nombres personales aplicada directamente a estos para reducirlos a una forma común [1]. Dado que los sistemas actuales han sido desarrollados originalmente para el idioma inglés, se hace indispensable realizar un estudio fundamentado a partir de la bibliografía existente, para ajustar el nuevo sistema a implementar de acuerdo con necesidades actuales del GINA.

1.2.1. Identificación de nombres personales. Ventajas de la similitud fonética.

Las técnicas de búsqueda de nombres personales incluyen los procesos a partir de los cuales se utilizan estas cadenas sustantivas como parte de una consulta para recuperar información asociada a personas naturales almacenadas en una base de datos. Existen dos situaciones posibles a partir de las cuales se puede obtener o no dicha información [1]:

- En la primera, la equiparación de los nombres es exacta, y en ese caso no se produce ningún problema.
- A diferencia de la segunda situación, donde puede que no se recupere información relevante al no ser capaz el sistema de establecer una equiparación exacta, debido a las variantes en la representación escrita de los nombres personales utilizados durante la construcción de la consulta y los que están incluidos en los registros de la base de datos.

Estos problemas de equiparación de nombres se podrían solucionar por medio de la aplicación de programas de comprobación ortográfica (spelling ckeckers), encargados de verificar los errores y corregir las variantes utilizando incluso diccionarios en los que se almacenarían las formas correctas. Dentro de los sistemas de corrección de errores, se pueden emplear básicamente dos planteamientos, los cuales se describen en los siguientes epígrafes.

1.2.1.1. Identificación por equiparación exacta.

Este procedimiento también conocido como “exact matching” consiste en la creación de dos diccionarios, uno con las formas correctas y otro con las variantes. Sin embargo, con este método sólo se corregiría una pequeña porción de variantes (que serían aquellas que previamente hubiesen sido almacenadas en el diccionario correspondiente).

1.2.1.2. Identificación por equiparación aproximada.

Dentro de este segundo procedimiento se aplican básicamente dos métricas:

- Medidas de similitud de cadenas.
- Medidas de similitud fonética.

Las medidas de similitud de cadenas (similarity measures), se basan generalmente en la minimización de la distancia, o en la maximización de la similitud entre las entradas del diccionario y las variantes. Para calcular este coeficiente de similitud, una medida muy conocida es [1]:

$$CS = \sum_{i=1}^{n} G$$

Donde *CS* es el coeficiente de similitud expresado a partir de la sumatoria de los *n-grafemas* comúnmente compartidos entre ambas cadenas.

Otra medida de similitud conocida es la distancia de edición (edit-distance) [1]:

$$ED = \sum_{i=1}^{n} S + \sum_{i=1}^{m} I$$

Donde *ED* es la distancia de edición expresada a partir de las sumatorias de *m-inserciones* y *n-supresiones* de caracteres necesarios para transformar una cadena en otra.

No obstante, es bien conocido que los programas de corrección ortográfica basados en algunas de las medidas anteriores no funcionan bien cuando se trata de comprobar la ortografía de los nombres propios.

La falta de normalización de este tipo de cadenas provoca la existencia de diversas limitaciones en las entradas de los diccionarios, e incluso la no disponibilidad de este tipo de recursos, debido a que la tarea de su almacenamiento se vuelve impracticable [1]. Estas limitaciones se deben fundamentalmente a la gran diversidad de estructuras de este tipo de cadenas originadas a partir de factores históricos y culturales.

Por su parte, las medidas de similitud fonética se basan en la asignación de la misma clave o código fonético, a los nombres que se pronuncian de forma parecida, prescindiendo además del uso de diccionarios lo cual constituye una ventaja significativa. Estos sistemas de equiparación sonora son utilizados habitualmente para simplificar la búsqueda en las bases de datos cuando solo se conoce la pronunciación de un nombre propio en lugar de su transcripción exacta, partiendo de la suposición de que los nombres que comparten la misma clave se podrían considerar similares y se han utilizado principalmente en aplicaciones que involucran la identificación de nombres personales.

1.2.2. Normalización fonética de cadenas.

Se ha planteado, que la existencia de los algoritmos fonéticos obedece a la necesidad de recuperar información que tiene una semejanza sonora, pero: ¿Qué lo hace posible? El estudio de la fonética se centra fundamentalmente en como un sonido puede en algunos casos ser representado por más de un grafema o combinación de letras y un solo grafema o combinación de letras puede ser a la vez representado por dos o más sonidos. Esto explica por ejemplo, porque las silabas SE, CE y ZE, ortográficamente poseen un significado gramatical relevante cuando forman alguna palabra, pero fonéticamente representan un sonido semejante, por lo cual prescinden de una combinación de grafemas del alfabeto para su representación, utilizando en su lugar un signo o fonema[6].

Los algoritmos de codificación fonética responden a este principio, razón por la que se ajustan a un conjunto de reglas de homogenización de grafemas similares, definidas previamente mediante un estudio de las características fonéticas diferidas de cada idioma por separado. Estas cláusulas se denominan:

[6] Representa los signos lingüísticos.

reglas de normalización fonética de cadenas y su funcionalidad principal se basa en la simplificación y transcripción de una palabra a su máxima expresión fonética, que no es más que identificar cada grafema o combinación de letras y asignarle un fonema representativo o código que sea el mismo para la sílaba en cuestión y para todas las fonéticamente semejantes existentes dentro de un dominio dado [4]. El código liberado por el algoritmo al final del análisis individual de cada cadena, será entonces común para un grupo de palabras de pronunciación similar. La idea es que una búsqueda, por ejemplo, del apellido "Gómez", también encuentre otros apellidos fonéticamente cercanos, como: "Gámez" o "Gomes" [10, 11].

1.2.2.1. Conceptos fonológicos que fundamentan las reglas de normalización.

A continuación, se describen algunos conceptos que fundamentan la utilización de las reglas de normalización fonética del castellano, en las cuales debe basarse el algoritmo de codificación idóneo para el desarrollo de la solución.

Grafema: "es la parte más pequeña del lenguaje escrito. Se corresponde con una letra del alfabeto y en el caso de los dígrafos, con dos letras" [11].

- Ejemplo: A, B, C, (…).

Dígrafo: se define como cualquier combinación de dos grafemas que representan un solo sonido [11].

- Ejemplo: CH, LL, RR.

Fonología: se encarga de distinguir los fonemas de una lengua [3].

Fonemas: representan la unidad básica de los diferentes sonidos [11]. Son distinguidos como los "signos lingüísticos" (consonánticos o vocálicos), capaces de variar significados léxicos o gramaticales [3].

- Ejemplo: /b/, /s/, /t/.
- Ejemplos de palabras representadas por grafemas y fonemas:
  - escribir => /eskribir/.
  - que => /ke/.

El estudio fonológico del idioma en cuestión, infiere que el objetivo fundamental de las reglas de normalización fonética, se centra en la unificación de las diferentes combinaciones de grafemas a través

de fonemas representativos que unifiquen los sonidos similares resultantes. Estos fonemas, expresados mediante un conjunto de signos previamente definidos, son almacenados como llaves conjuntamente con los nombres personales en los registros de la base de datos destinados para este fin, de forma tal que cada búsqueda se realice a partir de estas llaves, devolviendo entonces el nombre correspondiente.

1.2.2.2. Definición del alfabeto a utilizar.

En el caso del tipo de codificación fonética a utilizar para dar cumplimiento al los objetivos propuestos, se considerará cada símbolo o letra por separado, de tal forma que los dígrafos no serán tomados como un símbolo por sí mismos, si no como la combinación de dos grafemas. Por tanto el alfabeto de la solución consta de 27 letras:

a, b, c, d, e, f, g, h, i, j , k, l, m, n, ñ, o, p, q, r, s, t, u, v, w, x, y, z.

1.3. Propuesta de solución.

En este epígrafe se describen los distintos apartados que conforman la propuesta de solución, comenzando por los filtros como procesos preliminares aplicados a cada nombre personal, con el objetivo de unificar cadenas y abstraerlas de signos de puntuación o marcas diacríticas. También se describirán cada una de las reglas fonéticas del castellano que justifican el filtro de normalización, así como la determinación y selección del algoritmo de codificación a utilizar.

1.3.1. Filtrado inicial de la cadena de entrada.

El primer proceso de filtrado por el cual transita la cadena de entrada está distribuido de la siguiente forma:

- **Mayúsculas:** Todos los caracteres de la palabra serán transformados a mayúsculas. El objetivo de este filtro es trabajar a lo largo de todo el proceso de filtrado y normalización, con letras mayúsculas, para eliminar cualquier posible discrepancia que pueda existir entre los distintos tipos de letra.

- **ASCII:** Se procesará la cadena de tal forma, que algunos de los caracteres ASCII que se consideren equivalentes a letras, se transformen en el grafema correspondiente, como se muestra en la Tabla 1.9. Su objetivo fundamental es el de eliminar todos los signos ortográficos que se aplican a las letras como: tildes, signos monetarios, etc. Además de sustituir algunos caracteres ASCII comúnmente usados para representar ciertas letras.

EQUIVALENCIA ASCII

Caracteres ASCII	Equivalencia
Á, À, Â, Ä, Ã, Å, Æ	A
É, È, Ê, Ë, €	E
Í, Ì, Î, Ï	I
Ó, Ò, Ô, Ö, Õ, Ø	O
Ú, Ù, Û	U
¢	C
Ð	D
£	L
Ç, $, Š	S
×	X
Ý, Ÿ, ¥	Y
Ž	Z

Tabla 1.9: Representación de los caracteres ASCII.

- **Símbolos y marcas diacríticas:** El objetivo de este filtro se resume en conservar la integridad gramatical de las palabras procesadas, mediante la eliminación de los símbolos y las marcas diacríticas como por ejemplo las diéresis de la U o tilde de la Ñ, ya que es muy común no utilizarlas.

- **Decodificación UTF-8[7]:** Las cadenas recibidas a partir de algunos formularios pueden presentar caracteres Unicode, como es el caso de los signos de puntuación, etiquetados a partir de cláusulas

[7] 8-bit Unicode Transformation Format.

del HTML[8] nativo, razón por la cual es necesario decodificarlas antes de aplicar cualquier otra transformación sobre ellas.

1.3.2. Reglas a utilizar para la normalización de cadenas.

Las reglas de normalización que serán descritas a continuación, forman parte del proceso más complejo de filtrado a realizar sobre las cadenas de entrada, las mismas están basadas en las reglas fonéticas del castellano. Cada una de ellas se comporta como un algoritmo de unificación donde todas las combinaciones de letras que representan un sonido, son convertidas a un formato cuyo objetivo se resume en:

- Unificar todas las palabras similares en cuanto a pronunciación, independientemente de que algunas hayan sido escritas con faltas de ortografía, o sometidas a errores tipográficos comunes, como los cambios de letras adyacentes, presentes en los teclados de los ordenadores. De esta forma, el sistema ofrece un soporte más completo, permitiendo ampliar el margen de tolerancia contra errores cometidos por los usuarios, durante la realización de alguna operación que implique la escritura de nombres personales.
- Transformar cada palabra en los fonemas que la componen, incluso disminuyendo el tamaño de las cadenas procesadas, esto se debe a que las vocales intermedias también son eliminadas considerando que una palabra se distingue de otra, por la vocal inicial y las consonantes que conforman sus sílabas. La reducción de los nombres personales partiendo de esta técnica, permite acelerar considerablemente los procesos de almacenamiento y búsqueda dentro de dominios relativamente grandes.

✓ Descripción de las reglas de normalización.

Todos los grupos de letras que producen el sonido /K/, serán unificados partiendo de las siguientes consideraciones [11]:

[8] **HyperText Markup Language** (en español: Lenguaje de Marcado de Hipertexto).

- La consonante C, seguida de las vocales A, O, U, o de las consonantes R, L.
- La combinación de grafemas QU, seguida de las vocales E, I.
- La consonante Q, seguida de las vocales A, O.
- La consonante Q, seguida de la vocal U, mientras esta no esté sucedida por las vocales E, I.

Letras	Fonemas
CA, CO, CU	KA, KO, KU
CR, CL	KR, KL
QUE, QUI	KE, KI
QA, QO, QU	KA, KO, KU

Tabla 1.10: Consideraciones de normalización para el sonido /K/.

De igual forma todos los grupos de letras que producen el sonido /S/, serán tratados y sustituidos como tal:

- La consonante C, seguida de las vocales E, I.
- La consonante X, siempre que se encuentre al inicio de una palabra, sucedida por cualquiera de las vocales.

Letras	Fonemas
CE, CI	SE, SI
XA, XE, XI, XO, XU	SA, SE, SI, SO, SU

Tabla 1.11: Consideraciones de normalización del sonido /S/.

Los sonidos formados por las sílabas GUA, GUE, GUI, GUO, incluyendo la vocal U con diéresis en los casos GÜE, GÜI, serán sustituidos por el fonema /W/, al igual que las combinaciones vocálicas UA, UE, UI, UO, que forman hiatos o diptongos en los inicios de palabras [12].

Inicio de palabras	Letras	Fonemas
UA	GUA	WA
UE	GUE, GÜE	WE
UI	GUI, GÜI	WI
UO	GUO	WO

Tabla 1.12: Consideraciones de normalización del fonema /W/.

La consonante R no sufre transformaciones significativas, dado que su clasificación fonética es la de vibrante simple /R/ o vibrante múltiple /RR/, razón por la cual solo se tendrán en cuanta las siguientes consideraciones [12]:

- La consonante R solo se transformará en vibrante múltiple /RR/, si se encuentra implícita entre los grafemas N y vocal, o cuando inicia una palabra.

Los grupos de letras que producen el sonido formado por el dígrafo /LL/, serán sustituidos como tal, conforme a la siguiente tabla [12]:

Letras	Fonemas
YA, YE, YI, YO, YU	LLA, LLE, LLI, LLO, LLU

Tabla 1.13: Consideraciones de normalización del sonido /LL/.

Cuando al inicio de una palabra, aparecen las combinaciones HIE, IO, el fonema correspondiente más cercano por el cual se sustituyen está formado por: /Y|E/, /Y|O/, respectivamente. Es necesario resaltar la característica recursiva que debe ser aplicada en este caso:

- Una vez realizada la sustitución correspondiente, se aplica el tratamiento de la regla definida para la consonante Y, siempre y cuando esté sucedida de una vocal.

Es necesario destacar que la letra H es una consonante que no refleja ningún fonema, por tanto será eliminada independientemente de la posición en que aparezca dentro de la cadena tratada. De la misma manera será suprimida la consonante Y, en casos particulares donde realice función de vocal [12]. Se considera además que los grupos de letras idénticas y consecutivas a excepción de los dígrafos LL y RR, se unifiquen en una sola, por ejemplo: (CC por C).

1.3.3. Grupos de unificación sonora de consonantes.

Las consonantes que se corresponden con todos los posibles fonemas del castellano, han sido agrupadas según sus semejanzas sonoras, con el objetivo de unificar aún más las cadenas una vez normalizadas, mediante la asignación de un código a cada uno de los caracteres que aparezcan reflejados dentro de alguno de estos grupos. Estos códigos o signos de unificación de fonemas, son mostrados a continuación [12]:

GRUPOS DE FONEMAS

Códigos	Grupos de similitud de fonemas correspondientes al alfabeto
0	A E I O U
1	S C Z
2	V B
3	LL Y
4	G J X
5	K Q C
6	P
7	D
8	R
9	L
B	M N
C	F
D	T
F	Ñ
G	W

Tabla 1.14: Grupos de unificación sonora de fonemas del alfabeto.

Los grupos de unificación sonora no se corresponden con las reglas de la fonética del idioma propiamente dicho, pero constituyen una técnica muy eficaz a la hora de correlacionar fonemas similares. La codificación realizada a partir de estos signos representativos, forma parte del último proceso de filtrado y normalización de cadenas, necesario para la liberación de la clave fonética como llave de cada nombre propio almacenado en la base de datos.

1.3.4. Selección del algoritmo idóneo.

Unificar los procesos de filtrado, normalización y codificación de cadenas no es tarea fácil, debido a que debe seleccionarse una estructura algorítmica idónea, que sea capaz de administrar de la manera más óptima posible cada uno los procedimientos, dependiendo de la palabra evaluada en ese momento.

Fue necesario realizar un estudio detallado de las reglas de normalización, debido a que son las que mayor peso aportan a lo largo del proceso de filtrado, lo cual ha permitido reunir un conjunto de requisitos que deben ser evaluados en cada algoritmo referenciado, con el objetivo de seleccionar el más idóneo para formar parte de la solución abreviada del problema plateado. A continuación son mostrados estos requisitos:

- ✓ **Requisitos del algoritmo ideal.**

1. Las reglas de normalización utilizadas deben estar basadas en las reglas idioma-dependientes de la fonética para las cuales fue desarrollado dicho algoritmo.
2. Debe ser capaz de normalizar todas las consonantes presentes en el alfabeto seleccionado, incluyendo los dígrafos LL y RR.
3. Exceptuando la vocal que se encuentre en la primera posición, las demás deben ser suprimidas del resto de la cadena de entrada.
4. La vocal que aparezca en la primera posición de la cadena debe ser normalizada mediante caracteres comunes, con el objetivo de unificar un mayor número de variantes de nombres personales fonéticamente similares. Por ejemplo: en los nombres “Elianny” y “Alianny”, si no se normaliza la vocal inicial, el algoritmo no los asocia como palabras fonéticamente cercanas.
5. Las apariciones consecutivas de caracteres deben ser eliminadas, a excepción de los dígrafos LL y RR.
6. El código liberado debe tener una longitud de seis caracteres, en caso de que la cadena reduzca considerablemente sus letras y no cumpla con la longitud acordada, entonces serán completados los códigos finales con el dígito cero.

- ✓ **Representación gráfica de los requisitos planteados.**

La gráfica que aparece a continuación representa el número cuantitativo de requisitos reunidos por cada uno de los métodos estudiados. Su objetivo es el reducir el rango de algoritmos posibles a evaluar para la para la solución planteada.

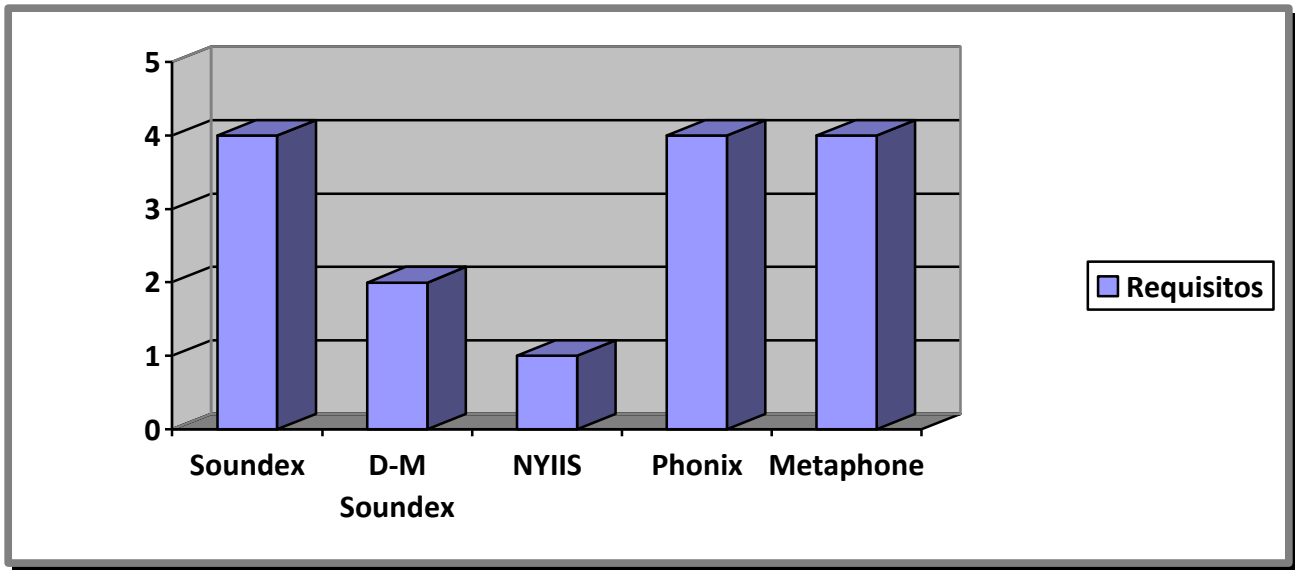

Figura 1.1: Representación gráfica de los requisitos reunidos por cada algoritmo.

- ✓ **Algoritmo Metaphone. Criterio de selección.**

La gráfica planteada cuantifica de igual forma los algoritmos: Soundex, Phonix y Metaphone, atendiendo al criterio de mayor cantidad de requisitos reunidos. Sin embargo existe una característica fundamental que cumple el algoritmo Metaphone, y que a su vez, no es común incluso para ninguno de los otros algoritmos planteados. Esta particularidad resalta la utilización de las reglas de pronunciación nativas de la fonética definida para cada idioma, y, en el caso de Metaphone para la lengua inglesa. La posibilidad que brinda este algoritmo, permite que sea fácilmente adaptable a las reglas de normalización fonética del castellano, razón por la cual pasará a formar parte de la solución objetiva del problema planteado.

1.3.5. Necesidad de implementar la solución planteada.

Mediante el estudio realizado sobre los diferentes sistemas de codificación fonética existentes, se determinó que ninguno satisface las necesidades actuales de la Aduana General de la República, sin embargo, estos aportaron algunos elementos imprescindibles para fomentar el desarrollo de la propuesta de solución basándose en las mejores prácticas reunidas hasta el momento. A continuación se muestran los fundamentos bajo los cuales será implementada la solución planteada:

- El tipo de normalización canónica de palabras utilizada por el algoritmo, debe basarse en las reglas del estudio fonológico del idioma castellano, con el objetivo de adoptar la pronunciación nativa del idioma en cuestión en lugar de utilizar la del inglés.
- Las reglas de normalización fonética identificadas, deben ser transcriptas al lenguaje de programación definido, fomentado de esta forma el objetivo de utilizarlas como librería persistente del motor de codificación.
- Existe una notable diferencia entre los grupos que unifican los diferentes sonidos vocálicos y consonánticos del alfabeto en ambos idiomas, razón por la cual debe redefinirse este orden, tomando en cuenta la semejanza fonológica del idioma castellano.

1.4. Conclusiones del capítulo.

De manera general, existe un número importante de Software en la Web que incorpora la tecnología de búsqueda y codificación fonética. A pesar de que la forma de empleo varía según el objetivo de la aplicación, su propósito final es el mismo: gestionar y recuperar de forma eficaz la información que modela cada negocio o entidad en particular.

Estas cuestiones denotan la diversidad y adaptabilidad de estos sistemas, cualidad fundamental tenida en cuenta para proporcionar una solución que se adapte a las características y necesidades específicas del Sistema GINA. De modo que el propósito general de este capítulo fue cumplido partiendo de los objetivos planteados anteriormente, los cuales se enmarcan en:

- Identificar todas las reglas de normalización fonética definidas para el idioma español, planteada la necesidad de transcribirlas al lenguaje de programación definido para el desarrollo de la solución.
- Definir un alfabeto que reúna los diferentes grupos basados en la similitud de los fonemas asociados a los sonidos de cada letra.
- Determinar cuáles de los algoritmos existentes se ajusta mejor para la asignación de claves fonéticas utilizando las reglas de normalización y codificación fonética identificadas.

2. CAPÍTULO II. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

En este capítulo serán descritos los métodos, tecnologías y herramientas utilizados para lograr la objetividad de la solución propuesta. Se modelarán también de forma consecutiva los aspectos del diseño basados en los diagramas de clases de diseño con estereotipos Web y consecuentemente se mostrarán los ajustes necesarios a realizar en el modelo físico funcional actual de la base de datos del Sistema GINA, para garantizar el correcto funcionamiento del motor de codificación y búsqueda fonética una vez implementado y puesto en práctica.

2.1. Tecnologías, métodos y herramientas utilizados.

2.1.1. Arquitectura del Sistema GINA.

La arquitectura de un Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos, los principios que orientan su diseño y evolución, así como el contexto en el que se implantarán. El objetivo principal de esta arquitectura es aportar elementos que ayuden a la toma de decisiones y al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la arquitectura del Software construye abstracciones, materializándolas en forma de diagramas [13].

La solución propuesta en este trabajo está enmarcada dentro de la arquitectura actual definida para el Sistema de Gestión Integral de la Aduana, la cual se representa gráficamente en la Figura 2.1 que aparece a continuación. Con el objetivo de lograr un mayor entendimiento acerca de la integración de la propuesta de solución con el actual dominio arquitectónico del Sistema GINA, se realizará una breve descripción que comprende los aspectos fundamentales de cada elemento presente en la arquitectura.

Es necesario aclarar además que esta arquitectura se basa en un patrón clásico muy efectivo para el diseño Web conocido como arquitectura MCV (Modelo-Vista-Controlador), que está formado por tres niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página Web que permite al usuario interactuar con ella.

- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo y en la vista.
- Debido a que el modelo se encarga de la abstracción lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes entre sí [14], el desarrollo de la solución a implementar estará enmarcado dentro del mismo.

Figura 2.1: Representación de la arquitectura del Sistema de Gestión Integral de la Aduana.

2.1.1.1. Uso de Symfony como Framework arquitectónico.

Este Framework ofrece una serie de características que lo hacen muy útil en el desarrollo de aplicaciones Web, algunas de estas se exponen a continuación:

- Facilita herramientas para desarrollar aplicaciones Web de alta complejidad.
- Adopta buenas ideas de otros proyectos, reutilizando el código existente cuando está disponible.
- El código desarrollado es fácil de mantener.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la Web.

- Separación de modelo, vista y controlador (implementación MCV).
- Independiente del sistema gestor de bases de datos.
- Fácil de instalar y configurar en la mayoría de las plataformas (garantizado para Windows y Linux).
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos [13].

2.1.1.2. Propel integrado a Symfony Framework.

"Propel, que también es un proyecto de Software libre, es una de las mejores capas de abstracción objetos-relacional disponibles en PHP5. Propel está completamente integrado en Symfony e incluso es su ORM[9] por defecto" [14]. Ambos desarrollan una combinación rápida y flexible en cuanto al acceso a la información de una base de datos. En la solución se utiliza Propel como capa de persistencia del modelo físico de datos.

2.1.1.3. Comunicación con la vista.

Es importante mencionar que la tecnología AJAX acrónimo de Asynchronous Java Script And XML (en español Java Script asincrónico y XML), permite realizar una comunicación asíncrona con el servidor en un segundo plano, de forma que se pueden hacer cambios sobre una página Web sin necesidad de actualizarse completamente [2].

Por su parte el JSON, acrónimo de "Java Script Object Notation", es un formato ligero para el intercambio de datos que se utiliza también para la comunicación asincrónica [2] entre el modelo y la vista. Consiste básicamente en un arreglo asociativo de Java Script donde se incluye la información de un objeto, de esta forma se minimiza el riesgo de que estos sean erróneamente modificados en la vista, garantizando así, una mayor seguridad [14]. También es necesario señalar que es el más adecuado para devolver grandes volúmenes de datos originados por transacciones o peticiones de búsqueda.

[9] ORM: Mapeo Objeto-Relacional.

2.1.1.4. La Web 2.0. Capa de presentación.

Este término no es más que una transición que ha ocurrido en los últimos años hacia el desarrollo de aplicaciones que funcionan a través de la Web, enfocada al usuario final. Su objetivo fundamental es reemplazar a las aplicaciones de escritorio por otras que ofrezcan colaboraciones y servicios igual de eficientes, pero a través de un navegador conectado a un servidor de aplicaciones [13].

El término Web 2.0 que utiliza la vista del Sistema GINA, está conformado por un entorno dinámico construido a través de un Framework llamado ExtJs, el cual define un conjunto de librerías Java Script que mejoran notablemente la calidad de las aplicaciones desde el punto de vista del usuario final, logrando un entorno gráfico más amigable a partir de un grupo de componentes desarrollados con este fin. También cuenta con validación de formularios, motivo por el cual será utilizado durante la realización de las pruebas al producto liberado por la implantación de la solución.

2.1.2. Visual Paradigm. Una Herramienta CASE.

Visual Paradigm se materializa como una herramienta CASE (Ingeniería de Software Asistida por Ordenador) profesional que soporta el ciclo de vida completo de desarrollo de Software: análisis y diseño orientado a objetos, construcción pruebas y despliegue. Sus características más acentuadas son:

- Soporta la versión UML 2.1.
- Soporta Ingeniería Inversa.
- Generación del modelo físico de datos.
- Gestiona de forma eficiente y organizada los artefactos generados durante la modelación de la solución.

2.1.3. Patrones de Diseño utilizados.

“Un patrón, en términos generales es una solución a un problema en un contexto dado, es recurrente, lo que hace que sea relevante para otras soluciones. Es una solución a un problema de diseño no trivial que es efectiva y reusable” [13].

Resumiendo, en términos generales, un patrón es un conjunto de información que proporciona una respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema dentro de un contexto, donde:

- Contexto: son las situaciones recurrentes a las que es posible aplicar el patrón.
- Problema: es el conjunto de metas y restricciones que se dan en ese contexto.
- Solución: es el diseño a aplicar para conseguir las metas dentro de las restricciones.

"Existen diversos patrones de diseño para desarrollar aplicaciones, estos se pueden agrupar en varias categorías como son los patrones GOF ("Gang of Four" o banda de los 4, debido a sus cuatro autores, los cuales fueron: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) y los patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades), estos últimos se consideran más que patrones propiamente dichos, como una serie de "buenas prácticas" de aplicación recomendables en el diseño de Software" [13]. A continuación se describirán los patrones utilizados para el diseño de la solución.

2.1.3.1. Patrón GOF implementado.

- ✓ **Patrón SINGLETON.**

Problemática: Obtener un punto de acceso global a una clase. (Instancia única).

Propósito: Asegurar que sólo exista una instancia de una clase específica en un sistema a desarrollar y la creación de un mecanismo de acceso global a dicha instancia.

Solución: La clase principal perteneciente al núcleo del motor que contiene el algoritmo de codificación fonética, proporciona un punto de acceso global mediante la llamada a la función encargada de la acción de codificación dada una cadena de entrada.

2.1.3.2. Patrones GRAPS implementados.

- ✓ **Patrón BAJO ACOPLAMIENTO.**

Problemática: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

Propósito: “Aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. Si no se usara este patrón los cambios en las clases afines ocasionarían cambios locales, generarían tantas dependencias que serían difíciles de reutilizar, además de que serían difíciles de entender por sí mismas al estar aisladas de sus dependencias” [13].

Solución: Las clase librería conformada a partir de la transcripción de las reglas de normalización fonética del idioma español, posee un bajo acoplamiento en este sentido, pues no presenta dependencia alguna con la librería definida para la lengua inglesa. Ambas son utilizadas por el motor de codificación de manera independiente, atendiendo a la configuración del idioma de las aplicaciones del Sistema GINA.

- ✓ **Patrón ALTA COHESIÓN.**

Problemática: ¿Cómo mantener la complejidad dentro de límites manejables?

Propósito: “Cada elemento dentro del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines y realiza trabajo excesivo por lo que si no se utilizara este patrón las clases fueran difíciles de comprender, difíciles de reutilizar, de conservar y las afectarían constantemente los cambios” [13].

Solución: Al igual que Symfony, el motor de codificación fonética agrupa las clases por funcionalidades que son fácilmente reutilizables. El uso de este patrón permite tener clases fáciles de mantener, de entender y reutilizar.

- ✓ **Patrón CREADOR.**

Problemática: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

Propósito: "Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento" [13].

Solución: Durante el diseño e implementación de la solución se le delegó la responsabilidad a la clase principal del codificador fonético, de crear los objetos referentes a las librerías de idiomas existentes según el tipo de codificación a realizar.

- ✓ **Patrón CONTROLADOR.**

Problemática: ¿Quién debería encargarse de los eventos de normalización y codificación fonética?

Propósito: "Facilitar la centralización de actividades, delegar las actividades en otras clases con las que mantiene un modelo de alta cohesión."

Solución: El funcionamiento del núcleo del motor de codificación fonética delega las responsabilidades principales a su clase controladora: CodificadorFoneticoGina, la cual contiene el algoritmo de codificación fonética destinado al procesamiento de cadenas, conjuntamente con los grupos de similitud de grafemas del abecedario definidos para idioma seleccionado. Además, se encarga de crear y controlar los objetos de las librerías que implementan cada una de las reglas de normalización utilizadas.

2.1.4. Sistema gestor de base de datos.

Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

La Aduana cubana ha adquirido experiencias positivas con Oracle usándolo desde el año 1997 [2].

2.1.5. Lenguaje de programación.

Como lenguaje de programación se utiliza el PHP en su versión 5.2.5. Conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones Web dirigidas a bases de datos. Este es un lenguaje de programación que se interpreta y ejecuta directamente en el servidor en el que está albergada la página Web, con lo que el visitante a la misma, únicamente recibe el resultado buscado por el código en el que está escrito. Es orientado a objetos y está ampliamente difundido en todo el mundo, presentando grandes volúmenes de documentación que son de gran ayuda para los desarrolladores [2].

2.2. Definición del modelo de diseño.

En el diseño se modela la solución para que cumpla con todos los requisitos, incluyendo los no funcionales del Sistema GINA y con otras restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario y tecnologías de gestión de transacciones, todas contenidas bajo una arquitectura previamente definida [13].

2.2.1. Diagramas de clases de diseño.

En el caso de las aplicaciones Web es más importante la modelación de la lógica, que los detalles de presentación, por tanto, para darle solución al diseño se utilizarán los diagramas de clase con estereotipos Web.

“Para lograr un nivel correcto de abstracción y detalle que permita obtener un resultado final de elevada calidad, se recomienda modelar todos los artefactos de la solución, es decir, conformar las clases, los enlaces entre estas, así como el contenido funcional de las mismas, los cuales conforman los resultados que serán mostrados en el navegador del cliente” [13].

Durante el desarrollo del diseño se contactará la documentación generada por los artefactos del proyecto, incluyendo la ayuda del arquitecto principal del mismo y el diseñador de la base de datos, con el propósito de lograr una correcta integración del motor de codificación y búsqueda fonética con las aplicaciones ya implementadas del Sistema GINA.

A continuación se muestra el diagrama de clases que describe el proceso general de codificación y búsqueda fonética a realizar por las aplicaciones del Sistema GINA.

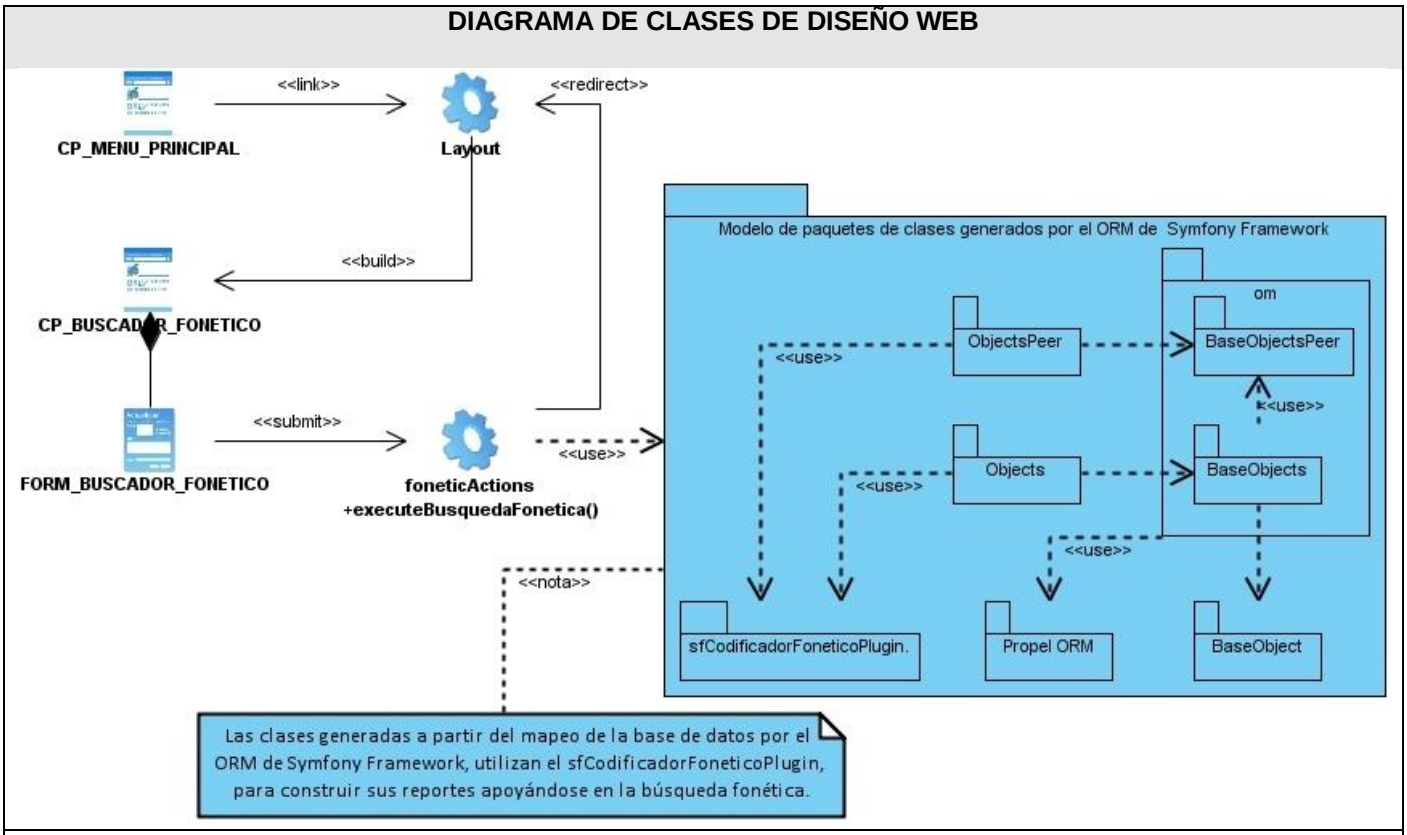

Figura 2.2: Representación general del proceso de codificación y búsqueda fonética.

La codificación y búsqueda fonética constituye una herramienta muy eficaz de validación que se activa durante la realización de una transacción o CRUD[10] parcial de información referente a personas naturales, con el objetivo de comprobar si ya fueron almacenadas anteriormente en la base de datos del sistema en cuestión.

[10] Acción de crear, actualizar o eliminar una información dentro de una base de datos.

El siguiente diagrama muestra una síntesis generalizada de este proceso en el cual la búsqueda fonética se utiliza como motor de validación de datos entre la información que se crea o actualiza y la existente en los registros de la base de datos, con el objetivo de realizar una seria de equiparaciones aproximadas basadas en la similitud de la pronunciación de los nombres personales, evitando así el duplicado accidental de dicha información.

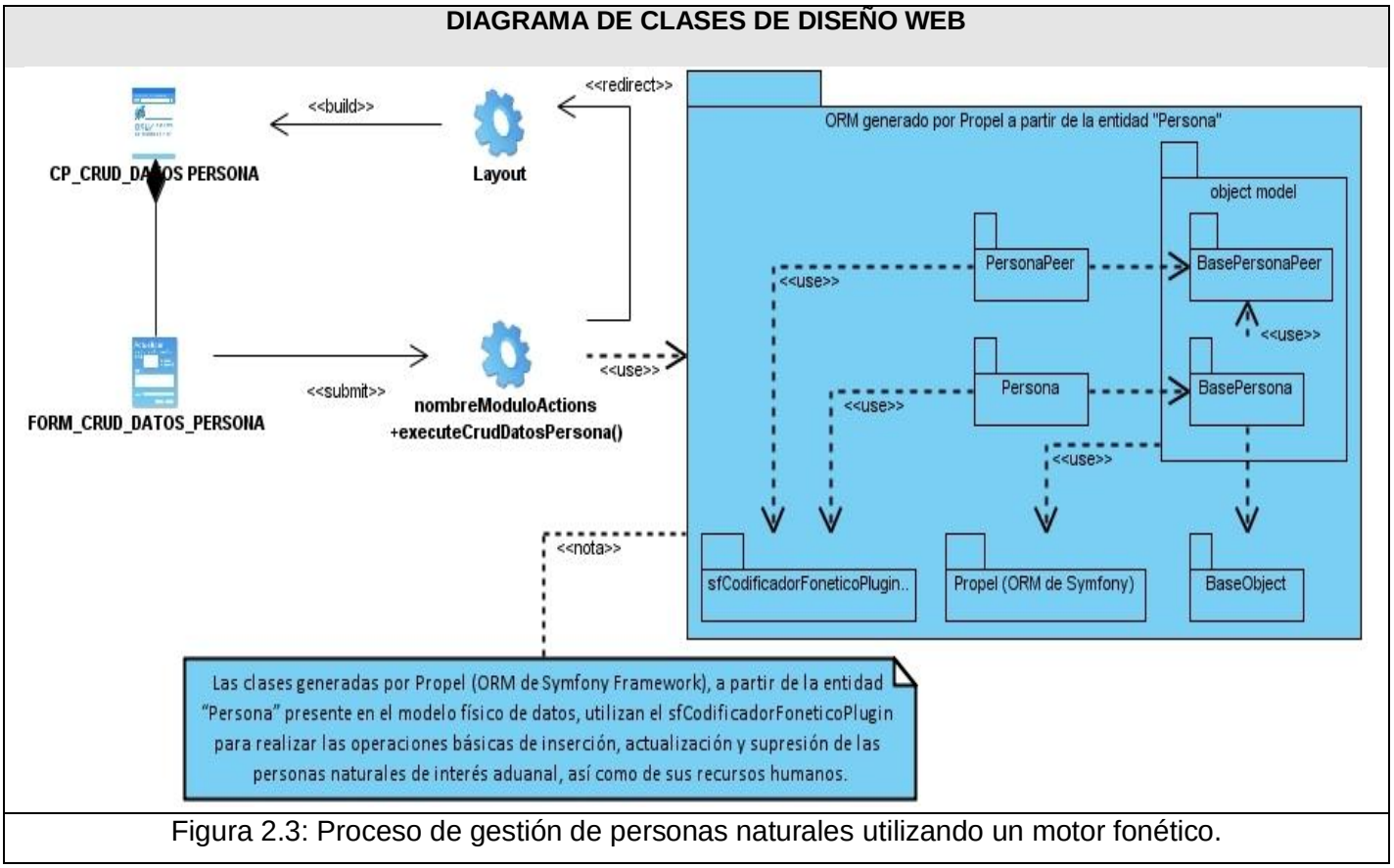

Figura 2.3: Proceso de gestión de personas naturales utilizando un motor fonético.

2.2.2. Clases base y clases personalizadas.

Symfony como Framework de desarrollo, permite una abstracción de la base de datos a partir de la generación de cinco clases por cada entidad del modelo físico de la misma. Para proporcionar una mejor comprensión de este modelo de objetos, se utilizará la clase persona como ejemplo ilustrativo ya que responde al propósito actual de este trabajo de diploma.

A partir de la entidad persona del modelo físico de la base de datos, el ORM del Framework genera las siguientes clases:

- Class Persona: esta clase aparece declarada pero con su cuerpo de código en blanco, se utiliza generalmente para agregar funcionalidades extras en caso necesario. Esta clase extiende de BasePersona.
- Class BasePersona: implementa todas funciones nativas que se pueden efectuar sobre una persona, tales como los métodos get() y set() de todos los atributos de la misma. Dado que su clase hija (Persona) hereda estas funcionalidades básicas, también pueden ser utilizadas en cualquier parte del código que se desee escribir sobre ella [13]:

```
$creaPersona->new Persona();

$creaPersona->setPrimerNombre("Carlos");

$creaPersona->setPrimerApellido("Romero");

$creaPersona->sabe();    //Esta función también es nativa para cada entidad a partir de la cual se genera. Permite salvar la información de la misma con solo citarla a través del objeto creado.
```

  Como muestra el ejemplo, esta clase facilita la creación y almacenamiento de una nueva persona en la base de datos sin necesidad de utilizar sentencias SQL[11]. De igual forma existe la función delete(): la cual permite eliminar una persona ya existente. Por otra parte se recomienda no agregar ni modificar el código predefinido de esta clase, pues si se determina en un futuro cambiar el motor de base de datos, entonces el nuevo modelo generado afectaría las modificaciones y se perdería cualquier función adicional agregada anteriormente. Esta clase extiende del núcleo de Symfony.

[11]Lenguaje de acceso a datos, utilizado para acceder a los registros de una base de datos.

- Class PersonaPeer: esta clase se utiliza de forma similar a la clase Persona, aunque está destinada fundamentalmente para desarrollar funciones estáticas que responden a reportes de búsqueda u actualización de información. Esta clase extiende de BasePersonaPeer [13].
- Class BasePersonaPeer: Las funcionalidades estáticas que se implementan en esta clase básica no deben ser modificadas bajo ninguna circunstancia, pues se utilizan para realizar consultas y actualizaciones en las operaciones dirigidas a la entidad física correspondiente, además de abstraer a los desarrolladores de construir consultas en código SQL.
- Class PersonaMapBuilder: este tipo de clases representan mapas estáticos de las entidades del modelo físico de la base de datos, utilizadas por Propel en tiempo de ejecución para la construcción de consultas SQL dinámicas partiendo de de cláusulas o criterios predefinidos por las clases del núcleo del Framework [14].

A continuación se muestra un diagrama que define la relación entre los paquetes de clases descritas anteriormente:

DIAGRAMA DE PAQUETES

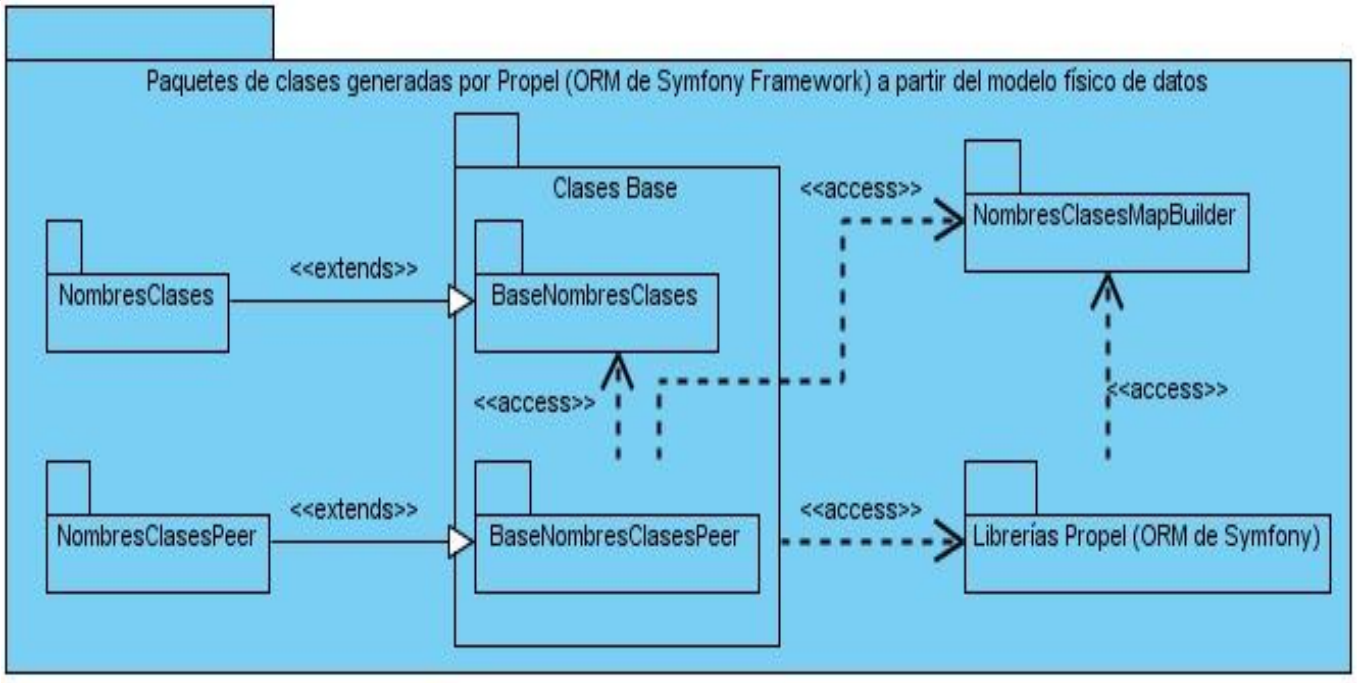

Figura 2.4: ORM generado a partir del modelo físico de datos.

2.2.2.1. Comportamiento del modelo de clases de objetos Base.

Es importante señalar que las clases de persistencia generadas a partir del mapeo de la base de datos, poseen las mismas relaciones que las entidades del modelo físico de datos, lo cual infiere a que todos los objetos que implementan dichas clases, estén también debidamente conectados. Esto supone sin lugar a dudas una notable ventaja para el diseño e implantación de la solución.

- ✓ **Diagramas de clases de objetos Base.**

Las clases Base como se ha referenciado en epígrafes anteriores, son las encargadas de crear, actualizar y almacenar la información de forma abstracta al motor de base de datos utilizado, además de proporcionar las relaciones correspondientes entre las clases generadas. Los siguientes sub-diagramas creados a partir de las memorias del proyecto GINA, permiten visualizar de forma clara como quedarían relacionados los objetos base tomando como premisa la agregación al modelo físico de datos de una entidad llamada Fonético, la cual se encargaría de almacenar los códigos fonéticos correspondientes al nombre de cada persona natural almacenada en los registros de la base datos, con el objetivo de realizar posteriores operaciones de búsqueda fonética y generar los reportes correspondientes.

SUB-DIAGRAMA DE CLASES DE OBJETOS BASE

Figura 2.5: Diagrama de clases de objetos Base perteneciente a la entidad Persona.

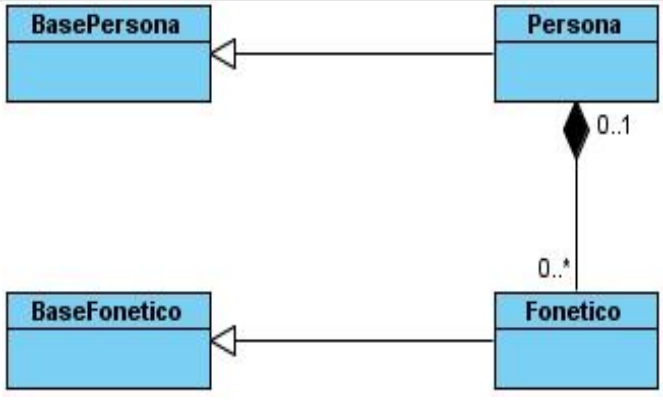

Figura 2.6: Diagrama de clases de objetos Base perteneciente a las entidades Persona y Fonético.

La relación de composición existente entre las clases Persona y Fonético expresada a través del diagrama sugiere que cada persona existente en la base de datos debe tener asociado un código fonético, de esta forma se crearían ambos objetos respectivamente y se almacenaría en la misma para ser utilizados posteriormente durante una búsqueda fonética. El próximo paso estaría enmarcado en redefinir los métodos set () de la clase Fonético, con el propósito de que almacenen el código equivalente a la cadena original tratada por la clase persona. El diagrama que aparece a continuación muestra la redefinición de estos métodos.

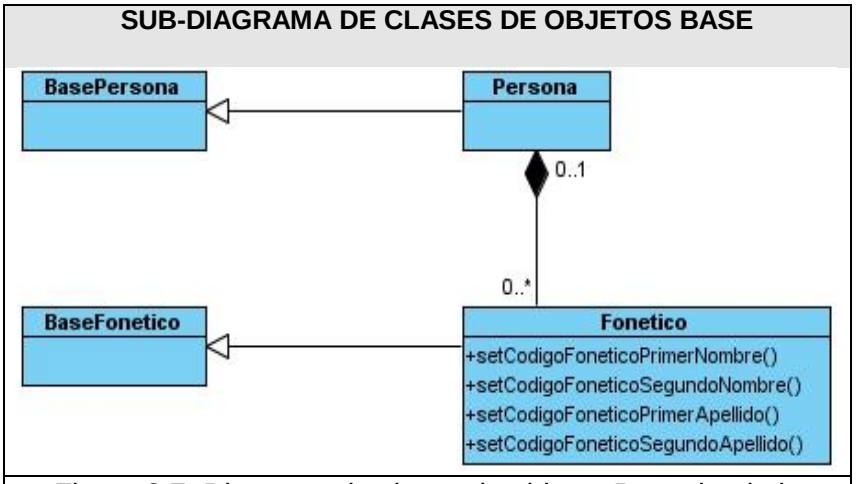

Figura 2.7: Diagrama de clases de objetos Base donde la clase Fonético redefine los métodos set().

Citando la explicación ofrecida en el epígrafe 2.2.2, se hace evidente subrayar que la redefinición de estos métodos no afectará el modelo generado siempre que se utilice la clase destinada para ese fin.

2.2.2.2. Comportamiento del modelo de clases de objetos Peer.

Las clases de objetos Peer son fundamentalmente utilizadas para crear y generar reportes de información que luego son enviados a la interfaz visual de la aplicación a través del controlador frontal de Symfony. Estas clases están equipadas con una serie de funcionalidades que permiten obtener más de un registro u objeto de la base de datos, para lograr este propósito implementan un método llamado doSelect() [14]. Esto permite consultar objetos de otras clases desde cualquier clase Peer, por ejemplo: a partir de la clase PersonaPeer se puede obtener el objeto Fonético mediante la línea de código FoneticoPeer::doSelect().

El primer parámetro requerido por el método doSelect() está conformado por un objeto de la clase Criteria, que es una clase para definir consultas simples sin utilizar código SQL, logrando de esta forma una abstracción de la base de datos. Es importante aclarar partiendo de la construcción de estas consultas, que un objeto Criteria vacío devuelve todos los objetos de una clase [14].

"Para las selecciones más complejas de objetos, se necesitan equivalentes a las sentencias WHERE, ORDER BY, GROUP BY y HAVING de SQL. El objeto Criteria dispone de métodos y parámetros para indicar todas estas condiciones" [14].

- ✓ **Diagrama de clases de objetos Peer.**

Al igual que ocurre con las clases Base, la clase PersonaPeer también será ligeramente modificada cuando se genere el nuevo modelo a raíz de la agregación de la entidad Fonético en el modelo físico de datos. Esta nueva relación de composición existente entre ambas clases será utilizada a favor de la construcción de del reporte de búsqueda fonética tal y como se ilustra en el siguiente diagrama.

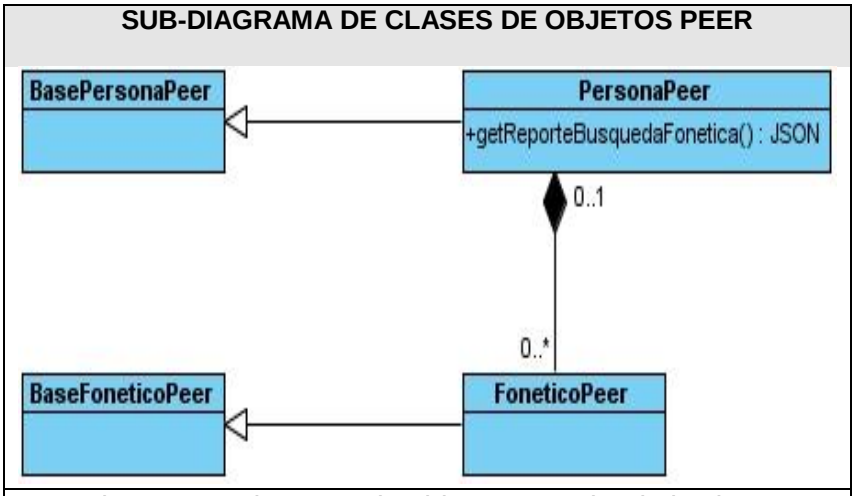

Figura 2.8: Diagrama de objetos Peer donde la clase PersonaPeer implementa la búsqueda fonética.

Debido a que la búsqueda fonética se realiza partiendo de la codificación de nombres personales, la implementación de dicho reporte tendrá lugar en la clase PersonaPeer. La devolución de este método será un JSON, que no es más que un estándar utilizado por Symfony para que un objeto creado a partir

de una consulta a la base de datos en el servidor sea convertido en un script que pueda ser interpretado por el cliente.

2.2.2.3. Otras ventajas de utilizar la clase Criteria.

Las funcionalidades de la clase Criteria permiten generar reportes mucho más potentes que una simple consulta SQL. “En primer lugar, se optimiza el código SQL para la base de datos que se utiliza. En segundo lugar, todos los valores pasados a Criteria se filtran antes de ser insertados en el código SQL, para evitar los problemas de SQL injection. En tercer lugar, el método devuelve un arreglo de objetos y no un resultset[12]. El ORM crea y rellena de forma automática los objetos en función del resultset de la base de datos. Este proceso se conoce con el nombre de hydrating” [14].

2.3. Diseño del codificador fonético.

Un motor de codificación fonética se basa en la utilización de un conjunto de reglas fonético dependientes de un idioma determinado, para llevar a cabo un proceso de normalización y codificación de cadenas a través de un algoritmo de codificación previamente definido, que agrupa los fonemas del alfabeto en distintos grupos de similitud fonética. Estos grupos se identifican a través de diferentes caracteres numéricos que conforman las cifras de los códigos generados. La clave fonética resultante será la misma para todos los nombres personales cuya pronunciación sea similar, lo cual permite que una búsqueda de esta naturaleza efectuada sobre un dominio dado (siempre que todos los códigos hayan sido generados pos el mismo motor de codificación), devuelva resultados que concuerden de forma sonora con la cadena de entrada.

2.3.1. Diagrama de clases que conforman el codificador fonético.

Los algoritmos de codificación fonética definidos para el idioma inglés han sido implementados de forma estructurada, lo cual dificulta su entendimiento además de limitar su reutilización para otros idiomas.

[12]Nativo del lenguaje PHP, se refiere a una matriz de datos generada directamente a partir de una consulta a la base de datos.

Utilizando las ventajas potenciales de la programación orientada a objetos (POO) como es el caso de PHP5 y Symfony es posible separar la parte lógica referente al algoritmo y la parte dinámica conformada por las reglas de normalización de cadenas (las cuales cambian en dependencia del idioma), en dos ficheros independientes, con vista a favorecer la reutilización de código, así como el acople para múltiples librerías PHP de reglas de normalización que puedan surgir.

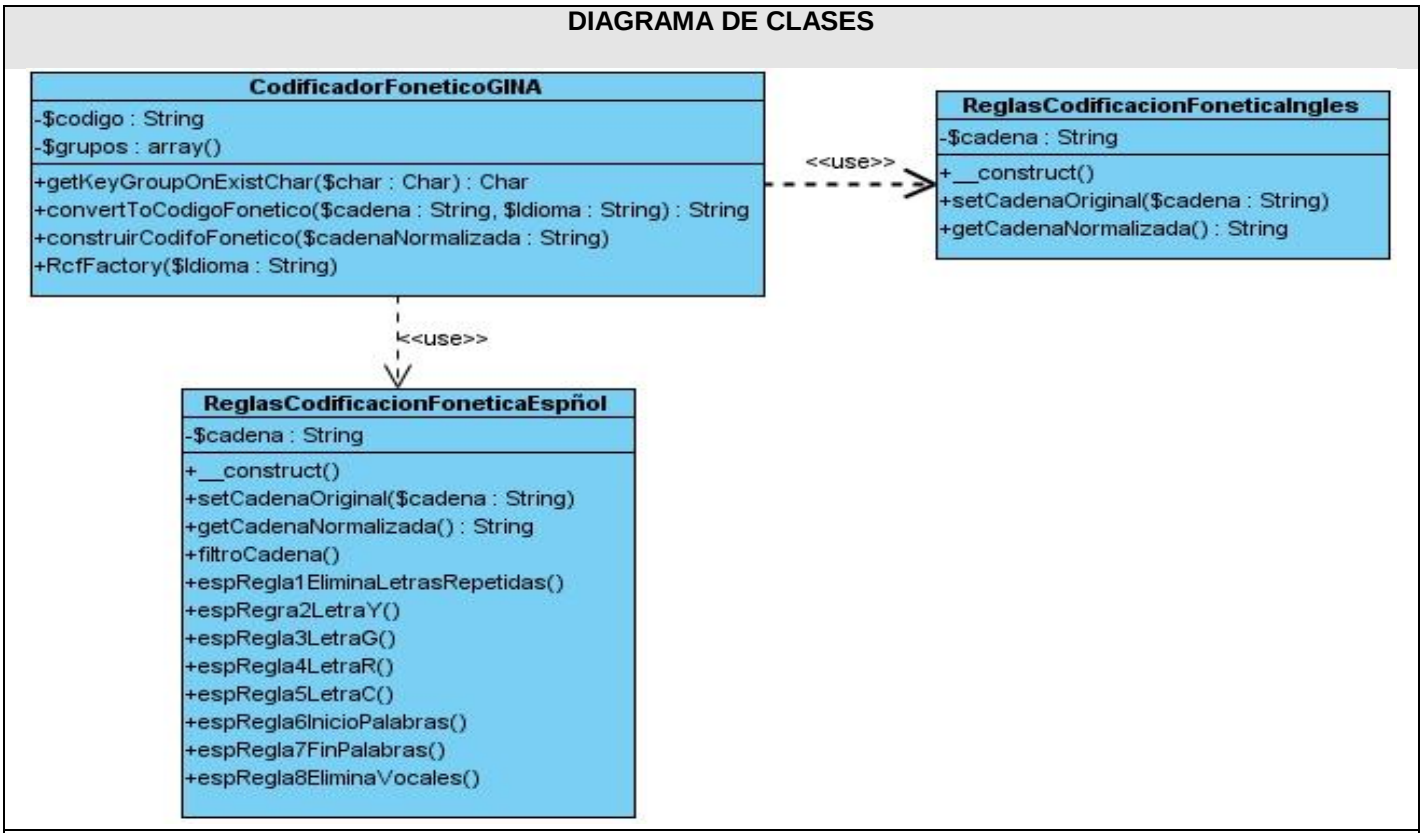

Figura 2.9: Diagrama de clases que conforman el codificador fonético.

2.4. Transformaciones a realizar en el modelo físico de datos.

Con la finalidad de materializar los aspectos recogidos en los epígrafes anteriores para dar solución a la propuesta planteada, se hace necesario realizar una ligera transformación en el modelo físico de datos utilizado hasta el momento por el Sistema GINA. Este reajuste brinda la posibilidad de almacenar en una entidad independiente, los códigos fonéticos generados a partir del procesamiento de datos realizado por el motor de codificación, además de relacionar cada uno con los nombres personales correspondientes.

2.4.1. Representación mediante el modelo entidad-relación.

A continuación se representa el cambio realizado en el modelo físico de datos a partir del diagrama entidad-relación de una porción a escala del modelo actual:

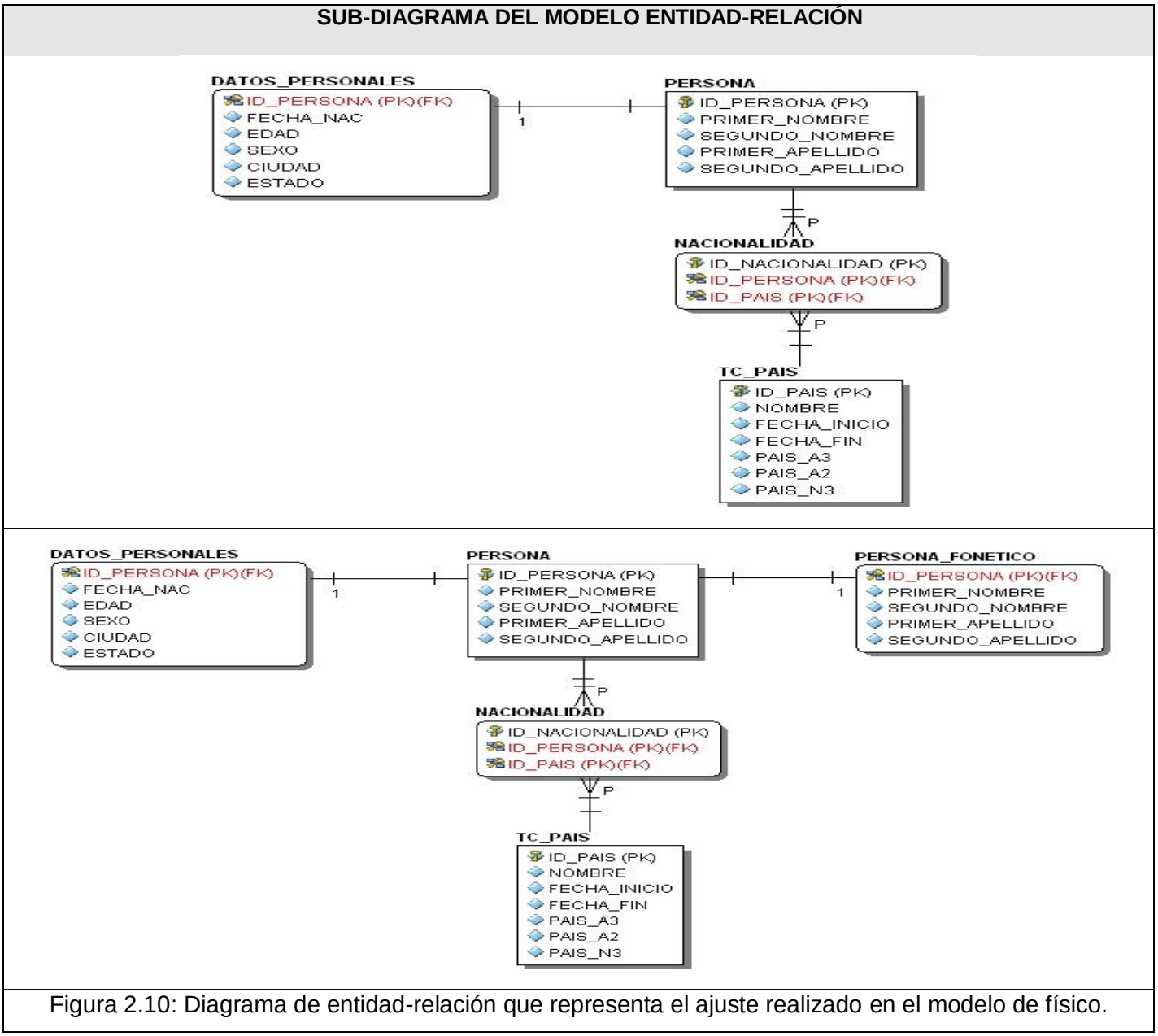

Figura 2.10: Diagrama de entidad-relación que representa el ajuste realizado en el modelo de físico.

2.5. Conclusiones del capítulo.

En este capítulo se definieron las características arquitectónicas de dominio específico del Sistema GINA, bajo las cuales se enmarca el diseño e implementación del la propuesta de solución. También se describieron las principales tecnologías, métodos y herramientas utilizados para dar soporte al desarrollo del sistema de codificación fonética, mediante las cuales quedó de establecida una línea de diseño donde se definieron los aspectos necesarios para el funcionamiento del algoritmo y la utilización de las reglas de normalización de forma independiente, cumpliendo de esta forma con los patrones de diseño pertinentes. Además, fueron descriptos los cambios realizados en el modelo físico de datos con el objetivo de garantizar el correcto funcionamiento del motor de búsqueda.

3. CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA.

Durante el transcurso de este capítulo se abordan temas correspondientes al desarrollo de la solución, tales como las técnicas de optimización algorítmica para disminuir sus demandas computacionales, así como la organización e interacción de los componentes utilizados durante su construcción. También se notificaran los resultados de un conjunto de pruebas unitarias realizadas durante las diferentes fases de construcción, comprendidas entre la transcripción de las reglas de normalización canónica y el desarrollo del algoritmo de asignación de códigos fonéticos, además de una prueba de funcionamiento del motor durante los procesos de inserción, actualización y búsqueda, tanto en entornos controlados como de despliegue piloto.

3.1. Complejidad algorítmica. Optimización.

Los algoritmos de codificación fonética empleados en los sistemas de búsqueda, suelen ser exigentes en sus demandas computacionales si no son debidamente optimizados. Cuanto más larga y compleja sea la tarea a abordar, mayores serán dichos requerimientos, de modo que es fundamental plantear técnicas específicas para reducir estas exigencias, sobre todo si el objetivo final comprende la aplicación de esta solución para sistemas de reconocimiento en tiempo real como es el caso del módulo de control de personas de la AGR.

De acuerdo con la estructura general de cualquier sistema de codificación fonética, como la mostrada en la Figura 3.1, se pueden realizar optimizaciones específicas en cada uno de los procesos fundamentales que conforman esta estructura: normalización canónica, codificación fonética y búsqueda/gestión.

3.1.1. Optimización del proceso de normalización canónica.

Dado que el proceso de normalización canónica se basa en la aplicación de un conjunto de reglas fonológicas a una cadena que representa un nombre de persona, el objetivo de la optimización se centra fundamentalmente en la organización de las consecuencias con las que se efectúan estas transformaciones sobre la expresión dada, de forma tal que se ejecuten en orden descendente atendiendo al criterio de complejidad de las mismas, es decir, desde la regla más compleja hasta la más simple. El

siguiente ejemplo de normalización canónica ofrece una mayor perspectiva acerca del tipo de optimización realizada.

Normalización					
L	O	´	P	E	Z
L	O	P	E	Z	
L	P	Z			
L	P				

Tabla 3.1: Ejemplo de normalización canónica realizada sobre una cadena de entrada.

En efecto, tal y como se muestra en el ejemplo anterior, la regla de filtrado inicial de la palabra es la que posee mayor complejidad, seguidas de otras cuyo objetivo es realizar un conjunto de transformaciones destinadas a transcribir la cadena de entrada a una expresión fonética normalizada.

3.1.2. Optimización del proceso de codificación fonética.

Este proceso comprende el funcionamiento del algoritmo de asignación de códigos fonéticos, a partir de los fonemas unificadores de los grupos de similitud sonora de las consonantes del alfabeto definido. La finalidad de optimizar este procedimiento se basa en lograr el mayor rendimiento posible del algoritmo en cuestión, para ello es necesario separar sus partes reutilizables en funciones auxiliares, con el objetivo de disminuir su complejidad y aumentar la reutilización de código.

3.1.3. Optimización del proceso de búsqueda y gestión.

La analogía de este proceso comprende ambas acciones, puesto a que para la creación en la base de datos de nuevos registros referentes a personas naturales, se hace necesario en primera instancia realizar una búsqueda de las mismas para comprobar si han sido o no insertadas con anterioridad.

La técnica de optimización de este procedimiento requiere fundamentalmente del aumento de los campos o criterios de entrada para reducir los horizontes de búsqueda, pues en diversas ocasiones las devoluciones efectuadas mediante la realización de una búsqueda fonética son muy numerosas, razón por la cual se hace necesario filtrarlas partiendo de criterios basados en otros datos personales como: sexo, fecha o rangos de fechas de nacimiento, y la nacionalidad.

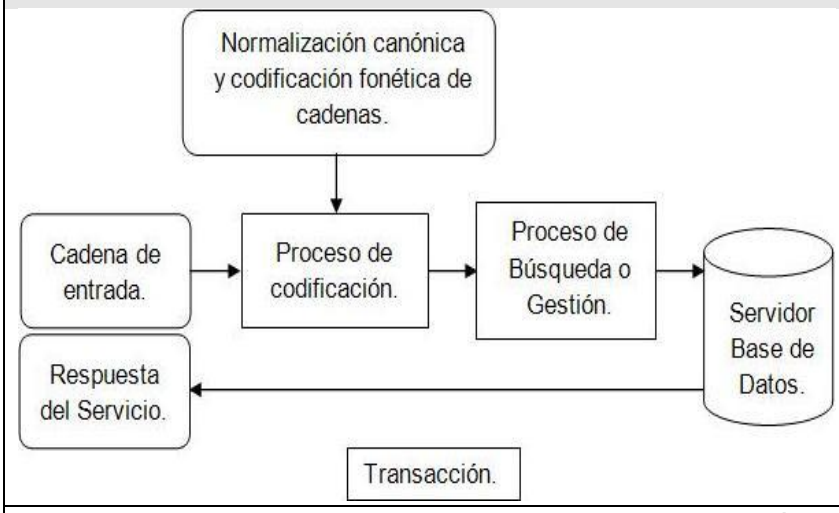

Figura 3.1: Estructura general de un sistema de codificación y búsqueda fonética.

3.2. Generalidades del modelo de despliegue.

Durante el desarrollo de la solución fueron tomadas en cuenta las especificaciones generales del modelo de despliegue utilizado para el Sistema GINA, el cual esta implementado siguiendo la arquitectura cliente-servidor. Tomando en cuenta que el cliente no depende de esta arquitectura para la realización de las operaciones o servicios que brinda el sistema, si se hace necesario que exista una total integración entre el motor fonético y las aplicaciones que lo utilizan, de modo que estas puedan acceder directa y globalmente a cada una de las funciones que este brinda.

3.2.1. Diagrama del modelo de despliegue.

El diagrama de despliegue que se muestra a continuación en la Figura 3.2, representa de forma general la ubicación del motor de codificación, el cual está integrado al Sistema GINA dentro del servidor de aplicaciones.

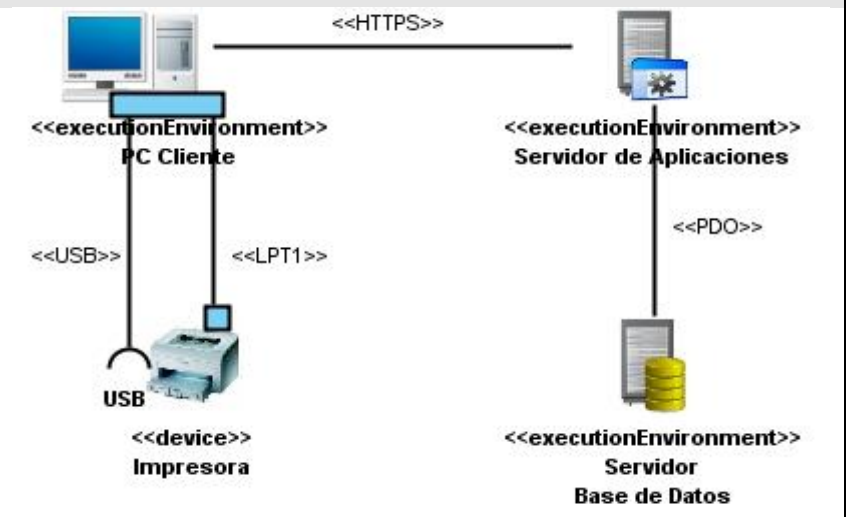

Figura 3.2: Diagrama de despliegue general del Sistema GINA.

Los servicios que brinda el sistema en cuestión están centralizados, lo cual significa que un único servidor responde a todas las operaciones realizadas por los analistas de la AGR. De igual forma, también se centralizan las funcionalidades generales del motor de búsqueda fonética, logrando una mayor eficiencia y rendimiento a la hora de retornar los resultados de las consultas.

3.3. Interacción entre los componentes del motor.

Un componente es una parte física reemplazable de un sistema que empaqueta su implementación y es conforme a un conjunto de interfaces a las que proporciona su realización. Algunos componentes tienen identidad y pueden poseer entidades físicas, que incluyen objetos en tiempo de ejecución, documentos, bases de datos, metadatos, etc. [11].

Los componentes que conforman la solución detentan dos características fundamentales:

- Empaquetan el código que implementa la funcionalidad principal, referida a la normalización y codificación canónica de cadenas que representan nombres de personas naturales.
- Ofrecen a su vez el punto de acceso global hacia dicha funcionalidad, por lo cual su instancia posee identidad propia.

3.3.1. Diagrama de componentes.

El diagrama de componentes que se muestra a continuación, representa la interacción entre las partes físicas que conforman el motor.

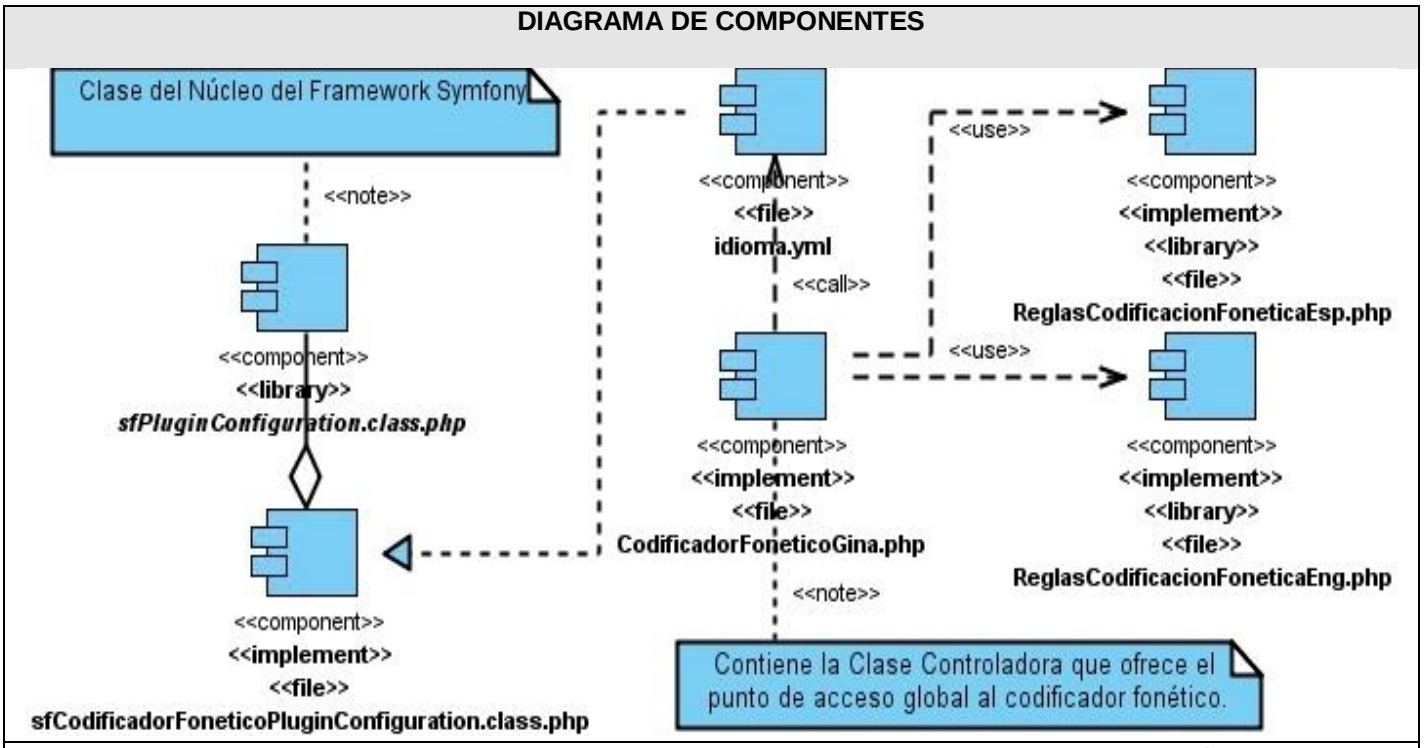

Figura 3.3: Diagrama general de componentes que conforman el codificador fonético.

3.4. Complementos del GINA. Forma de integración.

Un complemento o extensión de un sistema se define como una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Symfony implementa una interfaz de programación de aplicaciones o API (del inglés “application programming interface”), la cual permite que los proyectos puedan utilizar estos agregados como parte del propio Framework. Al encapsulamiento de estas funcionalidades externas, se le conoce en el lenguaje informático como: plugin o plug-in (del castellano “enchufado”) [14].

Las características de integración propias del Framework en este sentido, permiten que los plugins sean fáciles de instalar, activar y actualizar, sin necesidad de alterar el funcionamiento básico del proyecto desarrollado. Esta cualidad se torna fundamental para la adaptación del motor de codificación fonética al Sistema GINA.

3.4.1. Estructura de archivos de un plugin.

El directorio de un plugin se organiza de forma muy similar a la estructura de un proyecto. Los archivos contenidos dentro del mismo toman la forma adecuada para que Symfony pueda cargarlos automáticamente cuando sea necesario. A continuación se detalla la construcción del plugin contenedor del codificador fonético.

3.4.1.1. Aspectos para la construcción del plugin contenedor del codificador fonético.

El motor de codificación y búsqueda fonética debe estar activo durante todo el proceso de ejecución de las aplicaciones del Sistema GINA, por tanto, para que el plugin contenedor de esta solución funcione correctamente, es necesario seguir tres convenciones fundamentales exigidas por el Framework, las cuales son mostradas a continuación:

- La configuración del codificador fonético se incluye en el script de inicio del plugin (config.php). Este archivo se ejecuta después de las configuraciones de las aplicaciones del proyecto, por lo que Symfony ya ha iniciado cuando se procesa esta configuración [14].
- Las clases del núcleo del codificador fonético se cargan automáticamente de la misma forma que las clases que se guardan en las carpetas lib/ del proyecto.
- El nombre del plugin debe estar en correspondencia con el formato definido por Symfony, además de brindar información vital acerca de la función que realiza. Este formato exige utilizar el prefijo “sf” y el sufijo “Plugin”, durante el inicio y fin del nombre respectivamente. Ejemplo: sfNombrePlugin.

3.4.1.2. Personalizando archivos YAML para la configuración del tipo de codificación.

Dada la necesidad actual del Sistema GINA de brindar soporte para otros idiomas, la configuración del codificador fonético debe ser lo suficientemente dinámica y adaptable en este sentido, respondiendo, por supuesto, al conjunto de librerías contenedoras de reglas de normalización existentes, las cuales, hasta el

momento realizan normalizaciones canónicas para los idiomas inglés y español, siendo fácilmente extensibles para otros idiomas según se vayan incorporando las librerías que les den soporte.

Symfony utiliza por defecto el formato YAML (del inglés “Ain't Markup Language”) para la configuración, en vez de los tradicionales formatos INI y XML [14]. Este formato característico del Framework indica su estructura mediante la tabulación y es muy rápido de escribir, motivo por el cual fue definido dentro del directorio de configuración del plugin del codificador, un archivo llamado idioma.yml, con el objetivo de brindar acceso a la configuración del tipo de codificación a utilizar por el motor fonético.

3.4.1.3. Organización y distribución de los componentes del plugin.

Tomando como punto de partida los aspectos anteriores, el plugin contenedor del codificador fonético queda conformado de la siguiente forma:

- El fichero idioma.yml, encargado de proveer la configuración del idioma a utilizar durante el proceso de normalización canoníca de cadenas, queda contenido en el interior del directorio config/, con el objetivo de estructurar de forma organizada los archivos configurables, siguiendo además las exigencias descritas por el Framework en este sentido.
- El núcleo del codificador fonético conformado por el paquete de clases generado a partir del desarrollo de la solución, queda contenido dentro del directorio lib/, debido a que estas ofrecen el punto de acceso global al algoritmo utilizado por las diferentes aplicaciones del sistema durante el proceso de codificación fonética, motivo por el cual necesitan ser cargadas de forma automática una vez iniciado el GINA.
- Siguiendo el patrón definido por Symfony para la clasificación e integración de los plugins o extensiones de un proyecto, se determinó que el nombre del complemento contenedor de la solución debe ser sfCodificadorFoneticoPlugin[13], en correspondencia además con la función que este realiza.

[13] La ubicación del plugin se muestra en el Anexo 3.1.

3.4.1.4. Instalación y activación de sfCodificadorFoneticoPlugin.

Los procesos de instalación y activación del complemento sfCodificadorFoneticoPlugin se reducen solo a dos pasos fundamentales que son descritos a continuación:

- Acceder al directorio plugins/ del proyecto GINA y copiar el complemento definido en el interior del mismo.
- Verificar que el proyecto reconoce la nueva extensión, mediante la ejecución en consola de los comandos:
  - symfony cache:clear, utilizado para limpiar la memoria cache del proyecto, con el objetivo de cargar nuevamente todas a las aplicaciones existentes, incluyendo el nuevo plugin.
  - symfony plugin:list, para listar todos los plugins instalados, entre los cuales se incluye la nueva extensión.

Una vez instalado el nuevo complemento, se hace necesario comprobar que la función de instancia única que referencia al algoritmo de normalización canónica de cadenas, puede ser accedida desde cualquier parte de la aplicación mediante la línea de código: CodificadorFoneticoGina::convertToCodigoFonetico(). Este método recibe como parámetro la palabra a codificar fonéticamente y devuelve a su vez el resultado de tal acción.

3.5. Validación de la solución implementada.

La automatización de pruebas (del inglés “automated tests”) es uno de los mayores avances de la programación orientada a objetos. Concretamente en el desarrollo de las aplicaciones Web, las pruebas aseguran la validación de estas incluso para el desarrollo futuro de nuevas versiones. Este tipo de pruebas comparan un resultado con la salida esperada para una función o método específico. En otras palabras evalúan “asertos” (del inglés “assertions”), que son expresiones del tipo “true” o “false”, lo cual determina si la prueba tiene éxito o falla durante su realización.

Symfony incluye su propio Framework de automatización de pruebas llamado Lime, creado con el objetivo de facilitar la lectura de los resultados unitarios y funcionales de cada aplicación evaluada. Ofrece además

la facilidad de que cada prueba se realice en un entorno independiente para evitar interferencias de funcionamiento creadas por otras aplicaciones.

3.5.1. Metodología utilizada para la realización de las pruebas.

La metodología utilizada para la realización de las pruebas de validación del motor fonético, se conoce con el nombre de TDD o desarrollo basado en pruebas (del inglés “test driven development”), la cual establece que las pruebas deben escribirse antes que el código de la aplicación, de forma tal que el funcionamiento de cada uno de los métodos se centre en los resultados esperados. Se trata de una buena práctica recomendada para el desarrollo de soluciones estratégicas a corto plazo. Además, constituye un hecho innegable que si no se escriben las pruebas antes, se acaba sin escribirlas nunca [14].

3.5.2. Pruebas unitarias y funcionales realizadas.

Las pruebas unitarias[14] realizadas al codificador fonético aseguran que un único componente produzca una salida correcta para una determinada entrada, validando de esta forma que cada función trabaje correctamente para cada caso particular. Cada test automatizado se encarga de un único caso a la vez, lo que significa que un único método puede necesitar varias pruebas unitarias. En el caso del proceso de normalización canónica de cadenas, definido mediante las reglas que componen la librería utilizada por el algoritmo de codificación fonética, se realizaron un conjunto de pruebas especificas con el objetivo de evaluar el funcionamiento de cada una por separado.

Por su parte, las pruebas funcionales no solo evalúan la transformación de una entrada en una salida, sino que validan procesos completos, motivo por el cual requieren de un escenario o entorno de despliegue específico. La realización de pruebas piloto enmarcadas en un entorno real de trabajo, constituyen un escenario de evaluación experimental muy eficaz para la validación del motor de búsqueda y codificación fonética implementado para el Sistema GINA. En el siguiente epígrafe se ofrece un resumen detallado acerca de este tipo de validación realizada en la AGR.

[14] El documento de diseño de casos de prueba, referenciado en el Anexo 3.2, brinda mayores detalles acerca de las pruebas realizadas.

3.5.2.1. Pruebas pilotos realizadas en un entorno real de despliegue.

El Sistema GINA se encuentra actualmente desplegado en las estaciones de trabajo de la AGR, bajo un escenario similar al mostrado en la Figura 3.2, por lo que el motor de codificación y búsqueda fonética está siendo actualmente utilizado por las diferentes aplicaciones que requieren de sus prestaciones funcionales.

Este tipo de pruebas realizadas en entornos reales de ejecución, han permitido explotar al máximo las potencialidades del sistema de codificación fonética desarrollado a partir de la propuesta planteada, de forma tal que las inconformidades presentadas han sido resueltas para el peor de los casos esperados. Es importante destacar además, que la solución implementada cumple de forma óptima con las especificaciones de rendimiento requeridas por la dinámica actual de trabajo que exigen las aplicaciones del GINA, entre las que se distingue:

- La búsqueda fonética en tiempo real sobre bases de datos con registros de hasta más de un millón de personas almacenadas, realizadas por el módulo de control de personas de la AGR.

3.6. Conclusiones del capítulo.

En el presente capítulo fueron descritas las diferentes técnicas de optimización utilizadas para aumentar el rendimiento de los procesos realizados por el motor de codificación y búsqueda fonética implementado. También fueron explicados los procedimientos empleados durante la creación e instalación del complemento o plugin portador de la solución desarrollada para el Sistema GINA.

Por otra parte, fueron abordados aspectos puntuales comprendidos dentro de la validación de la solución implementada, la cual se realizó a partir de la aplicación de un conjunto de pruebas automatizadas mediante el Framework Lime de Symfony, diseñadas bajo la metodología TDD (desarrollo basado en pruebas). Se describe además la realización de pruebas pilotos bajo entornos reales de despliegue del sistema en cuestión, las cuales respaldan y validan de forma definitiva la propuesta materializada.

CONCLUSIONES GENERALES

El presente trabajo cumple de manara objetiva con las tareas de investigación propuestas, contribuyendo de esta forma a la resolución del problema planteado mediante la modelación de una la solución informática acorde a las necesidades y exigencias actuales de la AGR. De manera que la materialización final de la propuesta planteada cuenta con:

- Un alto grado de usabilidad por parte de los usuarios de la AGR, pues constituye una herramienta muy útil a la hora de realizar búsquedas en la base de datos cuando solo se conoce la pronunciación pero no la transcripción exacta de los nombres personales.
- Un óptimo rendimiento proporcionado por las técnicas de optimización utilizadas y confirmado además mediante las pruebas de despliegue realizadas en la aduana.
- Soporte no solo para realizar normalizaciones canónicas basadas en las reglas fonológicas del idioma español sino también de la lengua inglesa.
- Facilidad de integración con el actual Sistema GINA, debido al alto nivel de compatibilidad del plugin que encapsula el núcleo de la solución, con el Framework de desarrollo.
- Una marcada confiabilidad proporcionada a través de las distintas pruebas realizadas, incluso en entornos reales de trabajo.

RECOMENDACIONES

- Para versiones futuras del motor de codificación y búsqueda fonética del Sistema de Gestión Integral de la Aduana, se espera incorporar nuevos soportes de librerías que permitan la normalización canónica para otros idiomas además del español y el inglés.

- Se recomienda además implementar otros métodos basados en técnicas de cálculo por aproximación de distancias entre cadenas, con el objetivo de aportar cierto grado de inteligencia a la búsqueda fonética durante la realización de equiparaciones aproximadas de palabras con errores de traducción o transliteración.

BIBLIOGRAFÍA

1. Gálvez, C. (2006) *Identificación de Nombres Personales por medio de Sistemas de Codificación Fonética.* **Volume**, 13
2. Pérez, Y.R. and R.A.C. Basteiro, *Diseño del Módulo Control de Personas del Sistema Único de Aduanas.* 2009, Universidad de las Ciencias Informáticas: Ciudad de La Habana. p. 116.
3. Heredia, J.M. (2008) *La gramática: Conceptos Básicos.* **Volume**, 15
4. Cam, C.G. (2008) *Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos.* **Volume**, 8
5. Microsoft, C. *Novedades de la búsqueda empresarial (SharePoint Server 2010)*. 2009 2009-11-12 [cited 2010; 2010 Microsoft Corporation:[Available from: http://technet.microsoft.com/es-es/library/cc303422%28office.14%29.aspx.
6. Netmasters (2010) *Software de Gestión Empresarial*. **Volume**, 10
7. GenesisWorld (2010) *CAS genesisWorld: Vista General de Funciones*. **Volume**, 20
8. Barragán, M.A.R., *Un Método para Recuperación de Información en Documentos Orales basado en Codificación Fonética.* 2008, Instituto Nacional de Astrofísica, Óptica y Electrónica: Tonantzintla, Puebla.
9. Philips, L. *The Double Metaphone Search Algorithm*. 2000 [cited 2010; Available from: http://www.drdobbs.com/showArticle.jhtml;jsessionid=OUIIRCEO1N2THQE1GHPSKHWATMY32JVN?articleID=184401251.
10. Muñoz, F. *Algoritmos Fonéticos: Soundex*. 2009 [cited 2010; Available from: http://latecladeescape.com/w0/con-nombre-propio/algoritmos-foneticos-soundex.html.
11. Muñoz, J.A., E.L.S. Juan, and J.L.M. Carretero, *FUzzy MAtching System*. 2007, Universidad Europea de Madrid.: Madrid. España. p. 187.
12. Mestre, A.R. (1996) *Un alfabeto fonético del español para usos informáticos.* **Volume**, 237-244
13. Tamayo, L.H. and R.P. Herrera, *Información Adelantada de Pasajeros: Análisis y Diseño.* 2009, Universidad de las Ciencias Informáticas: Ciudad de La Habana. p. 99.
14. Potencier, F. and F. Zaninotto, *Symfony: la guía definitiva.* 2008.

GLOSARIO DE TÉRMINOS

Fonética: forma parte del objeto de estudio de la gramática y se encarga de definir y clasificar los sonidos del alfabeto [3].

Fonología: distingue los "fonemas" de una lengua [3].

Fonemas: son los "signos lingüísticos" (consonánticos o vocálicos) capaces de variar significados léxicos o gramaticales [3].

Errores Tipográficos: se entiende por estos tipos de errores los que son provocados al escribir o teclear palabras sin tener en cuenta la organización de sus letras o grafemas. Los errores más comunes son la sustitución de caracteres por otros fonéticamente similares o simplemente descuidos ortográficos.

Información Adelantada de Pasajeros y Tripulantes (API): "es la información obtenida de los datos del pasaporte o el visado de las personas que viajan a un país. Estos datos son transmitidos por medios electrónicos a las autoridades competentes del país de destino luego de la salida del vuelo, donde se analizan dichos datos para la gestión de riesgos antes de su llegada, con el fin de acelerar el despacho y evitar brechas de seguridad" [13].

Red SITA: Dada la importancia de la información que se maneja en las todas las aduanas con fines de seguridad nacional se torna necesario garantizar también la seguridad de la misma cuando es transmitida, por lo que en febrero del año 1949 se fundó la "Sociedad Internacional de Comunicaciones Aeronáuticas" o red (SITA) [13].

*Codificación **UTF-8***: (8-bit Unicode Transformation Format) es un formato de codificación de caracteres Unicode e ISO 10646 utilizando símbolos de longitud variable. UTF-8 fue creado por Robert C. Pike y Kenneth L. Thompson. Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes Web, o cuatro en ISO 10646 [11].

Application Programming Interface: se refiere al conjunto de funciones y procedimientos de Symfony (o métodos, en la programación orientada a objetos) que ofrecen una capa de abstracción entre las aplicaciones desarrolladas por el Framework y otras extensiones externas reutilizables [14].

Memoria cache de Symfony Project: Una de las técnicas disponibles para mejorar el rendimiento de una aplicación consiste en almacenar trozos de código HTML o incluso páginas enteras para poder servirlas en futuras peticiones. Esta técnica se denomina "utilizar caches" y se pueden definir tanto en el lado del servidor como en el del cliente [14].

Anexo 1.1

A continuación se muestra la tabla de codificación del método Daitch-Mokotoff Soundex, donde se pueden observar las reglas que se utilizan para codificar las diferentes letras, así como la ocurrencia de la modificación ortográfica de acuerdo a la secuencia de grafemas y la variación, si es al inicio de las palabras, antes de una vocal o en alguna otra situación.

Start of Before All				
Letter	Alternate Letter(s)	A Name	A Vowel	Other
AI	AJ, AY	0	1	N/C
AU		0	7	N/C
A		0	N/C	N/C
B		7	7	7
CHS		5	54	54
CH Try	KH (5) and TCH (4)			
CK Try	K (5) and TSK (45)			
CZ	CS, CSZ, CZS	4	4	4
C Try	K (5) and TZ (4)			
DRZ	DRS	4	4	4
DS	DSH, DSZ	4	4	4
DZ	DZH, DZS	4	4	4
D	DT	3	3	3
EI	EJ, EY	0	1	N/C
EU		1	1	N/C
E		0	N/C	N/C
FB		7	7	7
F		7	7	7
G		5	5	5
H		5	5	N/C
IA	IE, IO, IU	1	N/C	N/C
I		0	N/C	N/C
J Try	Y (1) and DZH (4)			
KS		5	54	54
KH		5	5	5
K		5	5	5
L		8	8	8
MN		66	66	66
M		6	6	6
NM		66	66	66
N		6	6	6
OI	OJ, OY	0	1	N/C
O		0	N/C	N/C
P	PF, PH	7	7	7
Q		5	5	5
RZ, RS Try	RTZ (94) and ZH(4)			
R		9	9	9
SCHTSCH, SCHTSH, SCHTCH		2	4	4
SCH		4	4	4
SHTCH	SHCH, SHTSH	2	4	4
SHT	SCHT, SCHD	2	43	43
SH		4	4	4
STCH	STSCH, SC	2	4	4
STRZ	STRS, STSH	2	4	4
ST		2	43	43
SZCZ	SZCS	2	4	4
SZT	SHD, SZD, SD	2	43	43
SZ		4	4	4
S		4	4	4
TCH	TTCH, TTSCH THS	4	4	4
TH		3	3	3
TRZ	TRZ	4	4	4
TSCH	TSH	4	4	4
S	TTS, TTSZ, TC	4	4	4
TZ	TTZ, TZS, TSZ, TS	4	4	4
T		3	3	3
UI	UJ, UY	0	1	N/C
U	UE	0	N/C	N/C
V		7	7	7
W		7	7	7
X		5	54	54
Y		1	N/C	N/C
ZDZ	ZDZH, ZHDZH	2	4	4
ZD	ZHD	2	43	43
ZH	ZS, ZSCH, ZSH	4	4	4
Z		4	4	4
N/C = not coded				

Figura 1.2. Tabla de codificación utilizada por el sistema Daitch-Mokotoff Soundex.

Anexo 1.2

El tipo de normalización canónica de cadenas utilizada por el algoritmo Metaphone, se basa en un estudio fonológico de la lengua inglesa, razón por la que a este sistema de codificación se le concede una gran importancia desde el punto de vista científico.

Consonante	Código	Condición
b	B	Excepto en el final de una palabra.
c	X	Si aparece como -cia-,-ch-
	S	Si aparece como -ci-,-ce-,-cy-
	K	En el resto de los casos.
d	J	Si aparece como -dgi-,-dge-,-dgy-
	T	En el resto de los casos.
g	J	Si anteceda a -i-,-e-,-y-
	N/C	Si aparece como -gh-
	K	En el resto de los casos.
h	N/C	Si precede a una vocal y no antecede a otra.
	H	En el resto de los casos.
k	N/C	Si precede a -c-
	K	En el resto de los casos.
p	F	Si aparece como -ph-
	P	En el resto de los casos.
q	K	Para todos los casos.
s	X	Si antecede a -h- ó aparece dentro de –sio-,-sia-
	S	En el resto de los casos.
t	X	Si aparece como -tia-,-tio-
	0	Si antecede a -h-
	T	En el resto de los casos.
v	F	En todos los casos.
w	N/C	Si no está precedida por vocal.
	W	Si está precedida por vocal.
x	KS	En todos los casos.
y	N/C	Si no está precedida por vocal.
	Y	Si está precedida por vocal.
z	S	En todos los casos.
N/C = not coded		

Figura 1.3. Reglas de normalización fonológica de cadenas utilizadas por el algoritmo Metaphone.

Anexo 3.1

En la figura se observa la ubicación del paquete sfCodificadorFoneticoPlugin, el cual posee una estructura de directorios similar en menor grado a la del proyecto generado por Symfony Framework.

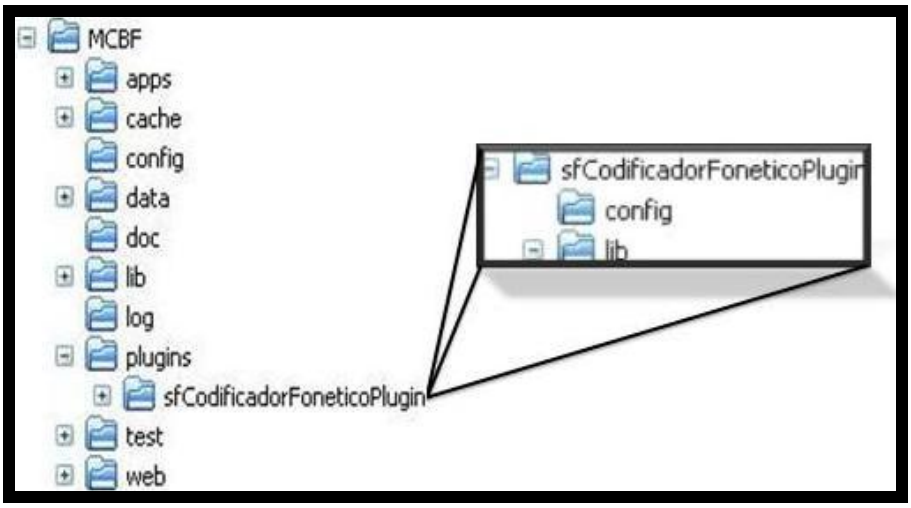

Figura 3.1: Ubicación del complemento contenedor del codificador fonético.

Anexo 3.2

Referencia al documento de diseño de casos de pruebas unitarias, realizado con el objetivo de evaluar el proceso de normalización canónica de cadenas utilizado por el algoritmo fonético.

Pruebas de normalización y codificación de palabras en castellano.doc