**SIMPLE**

*Sistema de Implementación de*

*Procesos Ligeramente Estandarizados*

MANUAL TÉCNICO Versión 1.0

*Gentileza de AGESIC  
Agencia de Gobierno Electrónico y Sociedad de la Información*

**TABLA DE CONTENIDO**

Tabla de Contenido ..................................................................................................................................... 2 Introducción ..................................................................................................................................................... 5 Resumen ...................................................................................................................................................... 5 Licencia ......................................................................................................................................................... 5 Ambiente de Ejecución ......................................................................................................................... 5 Plataforma, Framework o Lenguaje de Programación / BBDD ...................................... 5 Requisitos Técnicos ............................................................................................................................... 5 Instalación ........................................................................................................................................................ 6 Configuración del servidor .................................................................................................................. 6 PHP ............................................................................................................................................................ 6 Apache ..................................................................................................................................................... 6 Configuración de la plataforma ........................................................................................................ 6 Framework ............................................................................................................................................. 6 Base de datos ....................................................................................................................................... 6 Aplicación de parches ...................................................................................................................... 7 Carga de datos iniciales .................................................................................................................. 7 Habilitar módulo de búsquedas Sphinx .................................................................................. 7 Habilitación de soporte para múltiples subdominios ........................................................ 7 Permisos de carpetas ....................................................................................................................... 7 Configuración de operaciones recurrentes ........................................................................... 8 Configuración de Simple bajo HTTPS ..................................................................................... 8

Configuración de sesiones ............................................................................................................ 8 Ejecución inicial ........................................................................................................................................ 9 Administración de multiples cuentas ............................................................................................. 9 Modelo de datos ........................................................................................................................................ 10 Modelamiento de un Proceso ........................................................................................................ 10 Proceso ................................................................................................................................................. 10 Tarea ...................................................................................................................................................... 10 Paso ........................................................................................................................................................ 10 Formulario ........................................................................................................................................... 10 Campo ................................................................................................................................................... 10 Evento .................................................................................................................................................... 11 Acción .................................................................................................................................................... 11 Workflow de Tareas ....................................................................................................................... 11 Tipos de conexiones ...................................................................................................................... 11 Modelo Entidad - Relación .......................................................................................................... 12 Modelamiento de un Trámite .......................................................................................................... 12 Trámite .................................................................................................................................................. 12 Etapa ...................................................................................................................................................... 12

Dato Seguimiento ............................................................................................................................ 13

Estructura de Datos ................................................................................................................................. 14 Estructura de un Proceso ................................................................................................................. 14 proceso ................................................................................................................................................. 14 tarea ........................................................................................................................................................ 14 paso ........................................................................................................................................................ 16 formulario ............................................................................................................................................. 16

campo .................................................................................................................................................... 16 accion ..................................................................................................................................................... 18 Estructura de un trámite .................................................................................................................... 18 tramite .................................................................................................................................................... 18 etapa ....................................................................................................................................................... 18 dato\_seguimiento ............................................................................................................................ 19 Lenguaje de Expresiones ..................................................................................................................... 20 Variables Simple ................................................................................................................................... 20 Variables de Sesión Simple ............................................................................................................ 20 Variable Globales Simple ................................................................................................................. 21 Operaciones con expresiones ....................................................................................................... 21 Motor de expresiones ......................................................................................................................... 21 Extendiendo Simple ................................................................................................................................. 23 Nuevos tipos de campos .................................................................................................................. 23 Nuevos tipos de acciones ................................................................................................................ 23 Nuevas funciones de evaluación ................................................................................................. 24 Nuevas funciones de validación ................................................................................................... 25 Modificar y/o agregar nuevos tipos de documentos .......................................................... 25 Firma Digital Avanzada .......................................................................................................................... 27 Integración con HSM .......................................................................................................................... 27 Integración vía Webservice SOAP ......................................................................................... 27 Integración vía otros métodos ................................................................................................... 28

**Introducción**

**Resumen**

SIMPLE (Sistema para la Implementación de Procesos Ligeramente Estandarizados) corresponde a un sistema para la implementación de procesos electrónicos mediante un diseñador simplificado, desarrollado para entregar una solución flexible a instituciones públicas que deseen digitalizar sus trámites de forma amigable, rápida y sencilla.

**Licencia**

Este software esta licenciado bajo la licencia BSD-3.

**Ambiente de Ejecución**

Los usuarios deben ingresar mediante ambiente web, a través de la URL provista por el organismo.

**Plataforma, Framework o Lenguaje de Programación / BBDD**

-PHP, HTML, Javascript, CSS, MySQL

-Codeigniter 2.1.0

-Twitter Bootstrap 2.3.1

-jQuery 1.7.2

**Requisitos Técnicos**

• Apache 2.x (con modulo mod\_rewrite instalado y activado)

• PHP >= 5.3.x

• MySQL >= 5.5.x #Instalación

**Instalación**

**Configuración del servidor**

**PHP**

La instalación de PHP requiere tener el atributo short\_tags habilitado.

Para ello deben editar el archivo php.ini. Este archivo normalmente se puede encontrar en la ruta /etc/php5/apache2/php.ini.

Allí se debe habilitar el siguiente atributo:

short\_open\_tag=On

**Apache**

Se necesita habilitar el módulo rewrite de Apache para que funcionen las *friendly urls* de Simple.

Para habilitar el módulo se debe ingresar el siguiente comando en el terminal:

sudo a2enmod rewrite

También se necesita habilitar el uso de los archivos .htaccess. Para ello editar el archivo /etc/apache2/apache2.conf.

Buscar la sección <Directory /var/www/>. Dentro de ella, editar el siguiente atributo:

AllowOverride All

**Configuración de la plataforma**

**Framework**

• Cambiar el nombre del archivo de configuración del framework que se encuentra dentro de la carpeta "application/config" de "config.php.sample" a "config.php"

• Configurar el parámetro encryption\_key dentro del archivo config.php. Utilizar cualquier string aleatorio.

• Cambiar el nombre del archivo de configuración del envío de correos que se encuentra dentro de la carpeta "application/config" de "email.php.sample" a "email.php"

**Base de datos**

• Crear una base de datos para la plataforma

• Cargar la estructura inicial del archivo "sql/estructura.sql"

• Cambiar el nombre del archivo de configuración de base de datos que se encuentra dentro de la carpeta "application/config" de "database.php.sample" a "database.php"

• Editar el archivo "database.php" y actualizar los valores con los de su configuración, normalmente se deben editar sólo los campos "username, password, database"

**Aplicación de parches**

• Aplicar los últimos parches a base de datos situandose en la raiz de la aplicación y ejecutando el comando: php index.php migration migrate

**Carga de datos iniciales**

• Cargar los datos iniciales del archivo "sql/datos.sql"

**Habilitar módulo de búsquedas Sphinx**

• Instalar en el servidor el motor de búsquedas Sphinx (sphinxsearch.com)

• Cambiar el nombre del archivo de configuración de Sphinx que se encuentra dentro de la carpeta "sphinx" de "sphinx.conf.sample" a "sphinx.conf"

• Editar sphinx.conf incorporando los datos correspondientes a tu base de datos.

Para levantar el servidor de búsquedas, situarse en la carpeta sphinx de Simple. Luego ejecutar:

searchd

Para indexar los datos actualmente en la base de datos ejecutar:

indexer --rotate --all

**Habilitación de soporte para múltiples subdominios**

• Editar el archivo application/config/config.php y configurar la variable

$config['main\_domain']` con el dominio deseado.

Nota: Si se configura ejemplo.com como dominio principal. Cada cuenta de simple funcionara bajo el dominio nombredecuenta.ejemplo.com

**Permisos de carpetas**

Simple requiere que el servidor apache tenga permiso de escritura sobre las carpetas debajo de uploads/.

Si es que Apache no tiene esos permisos, se pueden setear mediante el siguiente comando.

Situándose en la carpeta raiz de Simple:

sudo chgrp -R www-data uploads sudo chmod -R g+rw uploads

**Configuración de operaciones recurrentes**

Simple requiere que ciertas operaciones se ejecuten cada cierto tiempo. Existe un set de operaciones que se debe ejecutar cada hora, y otra cada día.

Para ejecutar estas operaciones se debe ejecutar el siguiente comando en la raíz de Simple:

php index.php cron hourly php index.php cron daily

Si se desea configurar para que se ejecuten automáticamente, editar el archivo cron del sistema mediante:

crontab -l

Agregar las siguientes líneas:

# Se ejecuta cuando el sistema bootea

[@reboot](mailto:@reboot) php /ruta/a/simple/index.php cron hourly

[@reboot](mailto:@reboot) php /ruta/a/simple/index.php cron daily

# Se ejecuta todas las horas en el minuto 0.

0 \* \* \* \* php /ruta/a/simple/index.php cron hourly

# Se ejecuta todos los días a las 00:00

0 0 \* \* \* php /ruta/a/simple/index.php cron daily

**Configuración de Simple bajo HTTPS**

Si se desea que Simple funcione bajo el protocolo HTTPS, se debe configurar el siguiente parámetro de Simple ubicado en application/config/config.php

/\*

\* Servidor soporta HTTPS

\*/

$config['https'] = TRUE;

**Configuración de sesiones**

Por defecto, Simple almacena toda la información de sesión de los usuarios dentro de una cookie encriptada. Si se desea que la información de sesión se almacene en base de datos se deben seguir los siguientes pasos:

1- Crear la tabla de sesiones mediante la siguiente consulta SQL en la base de datos:

CREATE TABLE IF NOT EXISTS `ci\_sessions` ( session\_id varchar(40) DEFAULT '0' NOT NULL, ip\_address varchar(45) DEFAULT '0' NOT NULL, user\_agent varchar(120) NOT NULL,

last\_activity int(10) unsigned DEFAULT 0 NOT NULL, user\_data text NOT NULL,

PRIMARY KEY (session\_id),

KEY `last\_activity\_idx` (`last\_activity`)

);

2- Luego en el archivo application/config/config.php configurar el siguiente parámetro:

$config['sess\_use\_database'] = TRUE;

**Ejecución inicial**

• Acceder a http://nombrededominio/backend Usuario: admin@admin.com

Contraseña: 123456

**Administración de multiples cuentas**

• Acceder a http://nombrededominio/manager Usuario: admin@admin.com

Contraseña: 123456

**Modelo de datos**

**Modelamiento de un Proceso**

**Proceso**

• Es la unidad básica donde se almacena el modelamiento de un trámite

• Incluye toda la información del workflow del proceso

• Cada proceso corresponde a una fila en la tabla *proceso*

**Tarea**

• Corresponde a las diferentes etapas que se siguen para completar un trámite

• Son ejecutadas por algún usuario asignado

– La asignacion se puede definir como: ciclica, autoasignacion, manual o programada

• Esta compuesta por:

– Pasos (Ejecutados por usuarios)

– Eventos (Ejecutados en servidor automaticamente)

**Paso**

• Son las etapas que el usuario debe ejecutar para completar una tarea

• Cada paso tiene asignado un formulario

– Modo edición: El usuario debe ingresar input en el formulario

– Modo visualización: El formulario solamente esta disponible para ser visualizado

• Cada paso puede tener condiciones para ejecutarse o no

**Formulario**

• HTML forms donde el usuario ingresa la información requerida

• Están compuestos por distintos campos HTML

• Se pueden estructurar para tener un orden y una forma deseada

• Permiten insertar javascript si se desean comportamientos mas complejos

**Campo**

• Campo HTML para el ingreso de información

• Varios tipos

– textinput, textarea, select, check, radio, grid, etc

• Pueden tener condiciones para ser mostrados o no

• Se les puede asignar reglas de validación

– required, numeric, max\_length[num], etc

**Evento**

• Son las etapas que se ejecutan en forma automática en el servidor

• Cada evento tiene asignado una acción

• Se puede definir en que momento se ejecutaran

– Antes o después de una tarea

– Antes o después de un paso

• Se puede definir que ocurran bajo una condición

**Acción**

• Son operaciones que se ejecutan en el servidor

– Enviar correo

– Generar variable

• Permite almacenar en una variable el resultado de una expresión

– Consultar Webservice

• Permite almacenar en una variable el resultado de una llamada a un webservice

**Workflow de Tareas**

• Las tareas de un proceso siguen un workflow

• Las tareas están conectadas unas a otras

• La información de conexiones se almacena en la tabla conexión

• Cada conexión tiene una tarea origen, destino y condición con que se avanza

• Además hay distintos tipos de conexiones

**Tipos de conexiones**

• Secuencial: La mas básica, simplemente se avanza

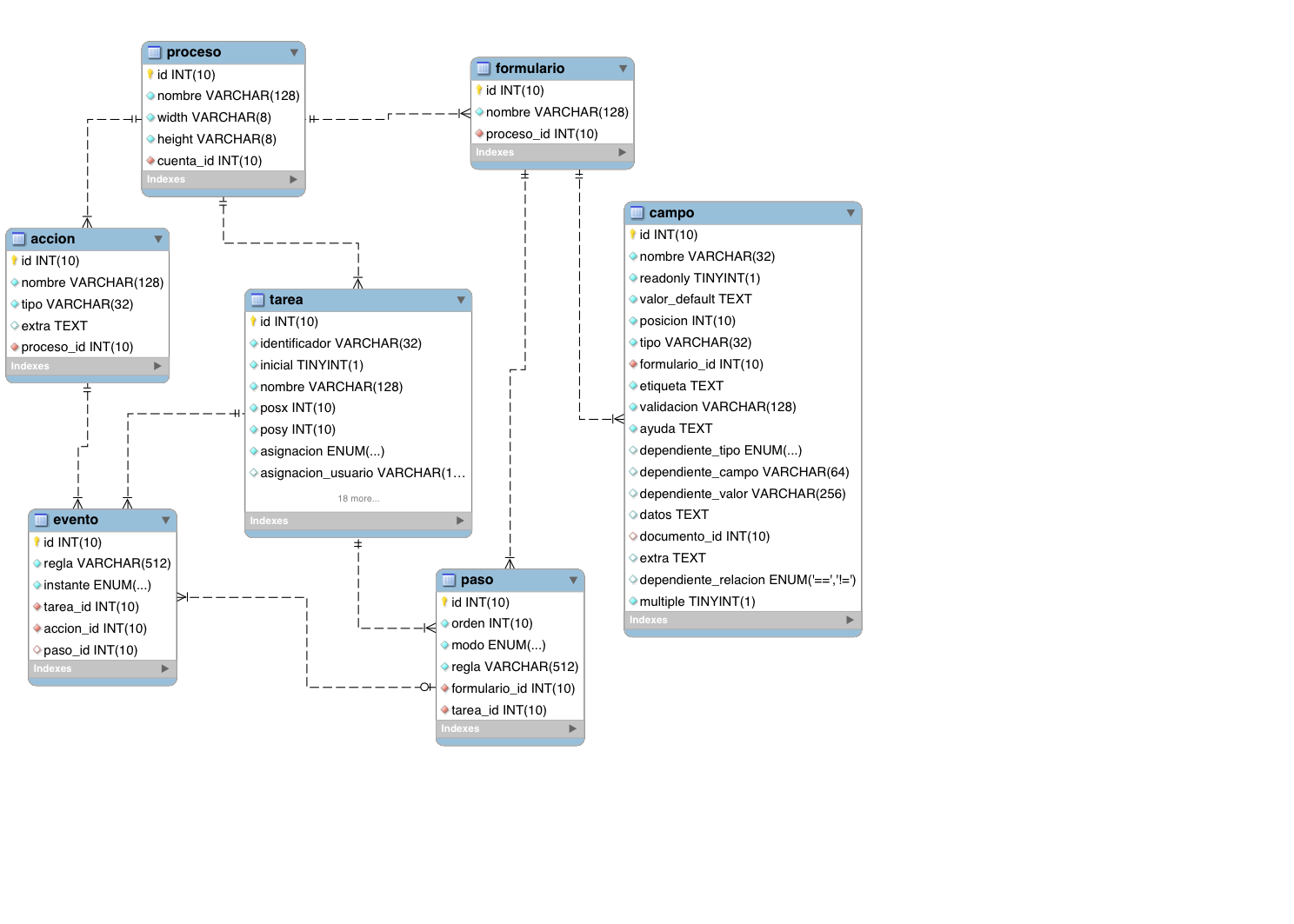
• Evaluación: Se avanza solo si se cumple la condición

• Paralela: Se avanza a múltiples tareas

• Paralela con evaluación: Se avanza a múltiples tareas si se cumple las condiciones respectivas

• Unión: Se une el flujo en una sola tarea (Se ocupa luego de que se dividió el flujo en paralelo)

**Modelo Entidad - Relación**



*Modelo Entidad - Relación*

**Modelamiento de un Trámite**

**Trámite**

• Es una instancia de un proceso

• Ej: Cada vez que un usuario inicia un proceso, se genera un nuevo trámite

• Se almacena

– Su estado: pendiente o completo

– Fecha de inicio

– Fecha de término

**Etapa**

• Corresponde a una instancia de una tarea

• Se almacena el usuario que esta asignado a ejecutar esa tarea

• Ademas se almacena

– Fecha de inicio

– Fecha de fin

• Los datos ingresados en cada etapa quedan almacenados en la tabla

*dato\_seguimiento*

**Dato Seguimiento**

• Se almacenan los datos obtenidos durante el tramite

• Cada dato tiene un nombre (Nombre de la variable)

• El valor es un campo JSON

• Se almacena la etapa donde fue ingresado el dato

**¿Qué ocurre cuando una misma variable se reescribe varias veces durante el trámite?**

• Cada dato queda en una fila distinta de la tabla *dato\_seguimiento*

• Esta fila queda relacionada a la etapa en que se modifico

• Asi se mantiene la historia de cada dato

**¿Cómo obtener el dato actual?**

• Se agrupan los datos por nombre

• Se ordenan los datos por etapa de mayor a menor

• El primero de la lista sería el dato mas actual

**Estructura de Datos**

**Estructura de un Proceso**

Un proceso contiene toda la información sobre como funciona un proceso en

Simple. Sus reglas, flujo, formularios, etc.

**proceso**

Esta tabla almacena la info principal perteneciente a un proceso creado en Simple

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id | int | Identificador del proceso |
| nombre | varchar(128) | Nombre del proceso |
| width | varchar(8) | Ancho del canvas donde esta dibujado el modelo del proceso |
| height | varchar(8) | Alto del canvas donde esta dibujado el modelo del proceso |
| cuenta\_id | int | Identificador de la cuenta a la que pertenece este proceso |

**tarea**

Un proceso esta compuesto por varias tareas. Una tarea equivale a un conjunto de acciones que ejecutara un usuario dado dentro del proceso.

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id | int | Identificador de la tarea |
| identificador | varchar(32) | Identificador que utiliza la libreria JS |
|  |  | que dibuja el proceso para identificar  cada tarea |
| nombre | varchar(128) | Nombre de la tarea |
| inicial | bool | Indica si es tarea inicial del proceso |
| posx | int | Posición de coordenada x dentro del canvas de dibujo del proceso |
| posy | int | Posición de coordenada y dentro del canvas de dibujo del proceso |
| asignacion | enum | Forma de asignación de usuarios a |
|  |  | esta tarea: ciclica, manual, auto-  servicio y usuario |

|  |  |  |
| --- | --- | --- |
| asignacion\_usuario | varchar(128) | En caso de que la asignacion sea a usuario, aquí se indica el id del usuario a asignar o la variable |
|  |  | SIMPLE que contiene este id |
| asignacion\_usuario\_notificar | bool | Indica si se debe notificar via email al usuario asignado a esta tarea |
| proceso\_id | int | Identificador del proceso al que pertenece esta tarea |
| almacenar\_usuario | bool | Indica si se desea almacenar el id del usuario que ejecuto esta tarea |
| almacenar\_usuario\_variable | varchar(128) | Si se desea almacenar el usuario |
|  |  | que ejecuto esta tarea, aquí se indica |
|  |  | como se debe llamar la variable  SIMPLE que almacena ese id |
| acceso\_modo | enum | Quienes tienen permiso para ejecutar |
|  |  | esta tarea: grupos\_usuarios, publico,  registrados y claveunica |
| activacion | enum | Modo de activación de esta tarea: si, no o entre\_fechas |
| activacion\_inicio | date | Si la activación es entre fechas, aquí se indica la fecha inicial |
| activacion\_fin | date | Si la activación es entre fechas, aquí se indica la fecha final |
| vencimiento | bool | Indica si esta tarea tiene un periodo de vencimiento |
| vencimiento\_valor | int | Cantidad de tiempo en la que la tarea se vence |
| vencimiento\_unidad | enum | Unidad de tiempo en que se mide el |
|  |  | vencimiento: D (días), W (semanas)  o M (meses) |
| vencimiento\_habiles | bool | Indica si se deben considerar solamente los días hábiles |
| vencimiento\_notificar | bool | Notificar via email cuando la tarea esté cercana a vencer |
| vencimiento\_notificar\_email | varchar(255) | Email al que se debe notificar cuando la tarea este próxima a vencer |
| vencimiento\_notificar\_dias | int | Cantidad de (dias,semanas,meses) |
|  |  | para que se cumpla el vencimiento  de la tarea |

|  |  |  |
| --- | --- | --- |
| grupos\_usuarios | text | Listado de grupos de usuarios, separados por coma, los cuales tienen permisos para ejecutar esta |
|  |  | tarea |
| paso\_confirmacion | bool | Indica si la tarea debe incluir al final |
|  |  | un último paso que confirme si uno  desea avanzar a la siguiente tarea |
| previsualizacion | text | HTML que permite previsualizar el |
|  |  | contenido de una tarea antes de  ejecutarla |

**paso**

Una tarea esta compuesta por una serie de pasos que el usuario ejecuta. Esta tabla almacena información sobre estos pasos a ejecutar.

Columna Tipo Descripción

id int Identificador del paso

orden int En que posición se ejecuta este paso modo enum En que modo se ejecuta el paso: edicion o

visualizacion

regla varchar(512) Condición para que se ejecute este paso formulario\_id int Identificador del formulario que se debe ejecutar en

este paso

tarea\_id int Identificador de la tarea a la que pertenece este paso

**formulario**

Columna Tipo Descripción

id int Identificador del formulario nombre varchar(128) Nombre del formulario

proceso\_id int Identificador del proceso al que perteneces este formulario

**campo**

Esta tabla almacena la información de campos correspondientes a un formulario.

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id | int | Identificador del campo |

|  |  |  |
| --- | --- | --- |
| nombre | varchar(32) | Nombre de la variable con la cual serán almacenados los datos ingresados en este |
|  |  | campo. |
| readonly | bool | Indica si el campo es de solo lectura |
| valor\_default | text | Texto con que se llena el campo en caso de no tener data |
| posicion | int | Posición del campo dentro del formulario |
| tipo | varchar(32) | Tipo de campo: Puede ser text, file, radio, checkbox, textarea, select, etc. |
| formulario\_id | int | Identificador del formulario al cual pertenece el campo |
| etiqueta | text | Texto que acompaña al campo explicando que es el campo. (Es el label del campo) |
| validacion | varchar(128) | Reglas de validación que se aplicarán al |
|  |  | campo dado un input de usuario. Las reglas  van separadas de el signo " |
| ayuda | text | Ayuda contextual sobre el llenado del campo |
| dependiente\_tipo | enum | Solo aplica si es un campo en el cual su |
|  |  | visualización depende de otro campo. Indica |
|  |  | que tipo de comparación hacer para  encontrar el match con los valores del otro campo: string o regex |
| dependiente\_campo | varchar(64) | Solo aplica si es un campo en el cual su |
|  |  | visualización depende de otro campo. Indica  en que campo hacer la comparación de valores. |
| dependiente\_valor | varchar(256) | Solo aplica si es un campo en el cual su |
|  |  | visualización depende de otro campo. Indica  el valor con el cual se debe hacer la comparación |
| dependiente\_relacion | enum | Solo aplica si es un campo en el cual su |
|  |  | visualización depende de otro campo. Indica  que tipo de comparacion hacer: igualdad  (==) o desigualdad (!=) |
| datos | text | Estructura JSON que almacena los datos |
|  |  | adicionales para desplegar un campo. Por  ejemplo, en el caso de un select, aquí |
|  |  | estaría el listado de opciones dentro del |

|  |  |  |
| --- | --- | --- |
| documento\_id | int | select  Si este campo es de tipo documento, este |
|  |  | identificador corresponde al documento a  desplegar en este campo |
| extra | text | Estructura JSON que contiene configuración adicional que pudiera requerir este campo. |

**accion**

Esta tabla almacena las acciones y sus configuraciones que un proceso ejecutara en los momentos correspondientes. Estas acciones pueden ser de distintos tipos: webservices, envios de correo, etc.

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id nombre extra proceso\_id | int varchar(128) text  int | Identificador de la acción  Nombre de la acción  JSON que incluye configuración adicional  Identificador del proceso al que pertenece esta acción |

**Estructura de un trámite**

Un trámite incluye información de un proceso realizado. Cada vez que se inicia un proceso, se crea un tramite nuevo que almacena la información sobre esa ejecución.

**tramite**

Se almacena información de un trámite.

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id | int | Identificador del trámite |
| proceso\_id | int | Identificador del proceso que corresponde a este trámite |
| pendiente | bool | Indica si este trámite ya finalizó o sigue pendiente |
| created\_at | datetime | Fecha y hora de creación de este trámite |
| updated\_at | datetime | Fecha y hora de última modificación de este trámite |
| ended\_at | datetime | Fecha y hora del fin de este trámite |

**etapa**

Cada vez que una avanza de una tarea a la siguiente de un proceso, se genera una nueva etapa en el trámite

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id | int | Identificador de la etapa |
| tarea\_id | int | Tarea correspondiente a esta etapa |
| usuario\_id | int | Identificador del usuario que esta asignado a |
|  |  | esta etapa. Si aun no hay un usuario  asignado, el campo queda en NULL |
| pendiente | bool | Indica si esta etapa ya finalizó o sigue pendiente |
| etapa\_ancestro\_split\_id | int | Identificador de la etapa anterior en el caso se |
|  |  | que se haya bifurcado el proceso en varios |
|  |  | flujos. Sirve para saber cuando el flujo se  debe volver a unir en uno solo |
| vencimiento\_at | date | Fecha en que esta etapa vence |
| tramite\_id | int | Identificador del trámite al que pertenece esta etapa |
| created\_at | datetime | Fecha y hora de creación de esta etapa |
| updated\_at | datetime | Fecha y hora de última modificación de esta etapa |
| ended\_at | datetime | Fecha y hora del fin de esta etapa |

**dato\_seguimiento**

Aquí se almacenan las variables Simple y su contenidos

|  |  |  |
| --- | --- | --- |
| Columna | Tipo | Descripción |
| id nombre valor etapa\_id | int varchar(128) text  int | Identificador de la etapa  Nombre de la variable  JSON con el contenido de la variable  Identificador de la etapa donde se modificó la variable |

**Lenguaje de Expresiones**

Te permiten implementar lógica de negocio en el modelamiento de los procesos. Ej:

• Condiciones para avanzar hacia una tarea

• Renderizado de documentos

• Renderizado de formularios

**Variables Simple**

Simple almacena variables obtenidas durante un trámite. El valor de las variables es un JSON.

Ej:

["hola mundo", "bla bla"] "Juan"

{"codigo": "UY", "pais": "Uruguay"}

Los inputs de los formularios almacenan sus resultados en variables Simple. También las variables pueden ser llenadas por otros métodos:

• Acción Generar Variable

• Acción Consultar Webservice

Las variables se obtienen en Simple mediante la sintaxis:

[@@nombreDeVariable](mailto:@nombreDeVariable)

**Variables de Sesión Simple**

Simple adicionalmente almacena variables de sesión. Corresponden a datos del usuario logueado y realizando el trámite.

Las variables de sesión disponible son:

• nombres

• apellido\_paterno

• apellido\_materno

• email

• tramite\_id

Las variables de sesión se obtienen en Simple mediante la sintaxis:

[@!nombreDeVariable](mailto:@!nombreDeVariable)

**Variable Globales Simple**

Simple adicionalmente almacena variables con la información de todos los trámites correspondientes a un proceso.

Por ejemplo, si la variable de un trámite se llama edad, la variable global edad obtendrá un array de todos los campos edad que ha completado la gente que ha hecho el trámite.

Las variables globales se obtienen en Simple mediante la sintaxis:

[@#nombreDeVariable](mailto:@#nombreDeVariable)

**Operaciones con expresiones**

El motor de expresiones de Simple permite incluso aplicar funciones sobre una variable.

Ej:

strlen([@@nombreDeVariable](mailto:@nombreDeVariable))

Se admiten todas las funciones PHP que se encuentre dentro de una Whitelist:

application/libraries/SaferEval.php

Se permiten también operadores lógicos PHP

==, >=, <=, !=, <,>

**Motor de expresiones**

El motor que calcula el resultado de las expresiones simple se encuentra en:

application/models/regla.php

Esta clase Regla tiene los siguientes métodos:

// Constructor: Asigna la expresión a evaluar public function \_\_construct($regla)

// Evalúa una expresión y da su resultado public function evaluar($etapa\_id)

public function getExpresionParaEvaluar($etapa\_id) Obtiene la expresión que será evaluada

Por ejemplo, este es el resultado si ejecuta el método evaluar con la siguiente expresión:

strlen([@@nombre](mailto:@nombre)) => 7 //[@@nombre](mailto:@nombre) = "Nicolas"

Este es el resultado si se ejecuta en cambio el método getExpresionParaEvaluar strlen([@@nombre](mailto:@nombre)) => strlen(“Nicolas”) //[@@nombre](mailto:@nombre) = "Nicolas"

**Extendiendo Simple**

**Nuevos tipos de campos**

1 Extender la clase Campo y almacenarla en:

• application/models

2 Editar el método factory de la clase Campo para agregar el nuevo Campo.

2 Reimplemento la función display($modo,$dato,$etapa\_id)

// Funcion que debe retornar el html con que se despliega el campo en el formulario

HTML.

// $modo => edicion : visualizacion

// $dato => El dato que esta almacenado actualmente y se quiere desplegar en el campo

// $etapa\_id (Opcional) => Identificador de la etapa en la cual se esta desplegando el campo

display($modo,$dato,$etapa\_id)

3 Opcionalmente puedo reimplementar las siguientes funciones:

// Campos adicionales en el formulario de ingreso de campos en editor de formularios backendExtraFields()

// Validaciones adicionales en el formulario de ingreso de campos en editor de formularios backendExtraValidate()

4 Editar los archivos de controladores y vistas necesarias para agregar a la interfaz gráfica la posibilidad de agregar estos nuevos campos:

• application/controllers/backend/formularios.php

• application/views/backend/formularios/editar.php

**Nuevos tipos de acciones**

1 Extiendo la clase Accion y almacenarla en:

• application/models

2 Editar el método setTableDefinition de la clase Accion para agregar la nueva

Accion

2 Reimplementar la función ejecutar(Etapa $etapa)

// Código que se debe ejecutar con la acción

//$etapa => Etapa en la que se esta ejecutando la acción. ejecutar(Etapa $etapa)

3 Reimplemento la función displayForm()

// Debe retornar el formulario que se debe desplegar para configurar la acción displayForm()

4 Reimplemento la función validateForm()

// Debe establecer las reglas de validación en los campos que configuran la acción validateForm()

**Nuevas funciones de evaluación**

El lenguaje de expresiones Simple tiene un Whitelist con las funciones permitidas

PHP que pueden ser ejecutadas.

El Whitelist puede ser editado en el archivo:

application/libraries/SaferEval.php

Si se desea agregar una nueva función PHP al Whitelist basta con agregarla al arreglo $this->allowedCalls

Si la función que se desea agregar es custom, es decir, no existe en la distribución de PHP, se deben seguir los siguientes pasos:

1- Crear un nuevo archivo helper en la carpeta application/helpers. El nombre debe seguir la nomenclatura MY\_foo\_helper donde foo es el nombre de tu helper.

2- En este archivo helper, escribes la función deseada. Por ejemplo, esta es una función que convierte una tabla JSON a HTML:

function matrix\_to\_html($matrix){

$html='<table class="table" >';

foreach($matrix as $row){

$html.='<tr>';

foreach($row as $data){

$html.='<td>'.$data.'</td>';

}

$html.='</tr>';

}

$html.='</table>';

return $html;

}

3- Editar el archivo application/config/autoload.php. En el arreglo de helpers agregar el nombre de tu helper.

4- Agregar el nombre de tu funcion al Whitelist en:

application/libraries/SaferEval.php

**Nuevas funciones de validación**

Simple trae una gama de funciones incoporadas para hacer validaciones sobre el input del usuario en campos de los formularios. Trae incorporado todo el set de funciones de validación ya incluidos en Codeigniter:

<http://www.codeigniter.com/userguide2/libraries/form_validation.html#rulereference>

Ademas Simple incorpora otras funciones de validación custom como: Función Parametro Descripción

alpha\_space string Valida si el string tiene solo caracteres alfa- numéricos y/o espacios.

is\_unique valor => string, json => json

Función que chequea si el valor es único. Es decir, no se encuentra dentro del conjunto json.

Si uno desea incorporar funciones de validación adicionales se debe editar el archivo:

application/libraries/MY\_Form\_validation.php

En este archivo se debe incorporar la función de validación que se desea.

La función recibe como primer parámetro el string con que se lleno el campo del formulario. A continuación va cualquier otro parametro adicional que se requiera.

La función debe retornar true si la validación es correcta y false en caso contrario.

**Modificar y/o agregar nuevos tipos de documentos**

Simple viene incorporado con dos tipos de documentos:

• **Blanco**: Documento en blanco sin ningún formato adicional. El usuario modelador de trámites puede incorporar HTML como contenido.

• **Certificado**: Documento con formato de certificado. Incorpora un header y footer con un diseño gráfico de certificado. Tiene elementos distintivos como logos, validadores QR, estampillas y firmas.

El template de cualquiera de estos certificados puede ser editado modificando los templates existentes:

• application/libraries/blancopdf.php

• application/libraries/certificadopdf.php

Ambas son clases que extienden de TCPDF. Ambas reimplementan los siguientes métodos:

// Constructor. Se pasan todos los parametros necesarios para el renderizado. function \_\_construct($parametro1, $parametro2, ..., $parametron)

// Se genera el contenido que ira en el header del documento (El header se repite en todas las páginas)

public function Header()

// Se genera el contenido del cuerpo del documento public function Content()

// Se genera el contenido que ira en el footer del documento (El footer se repite en todas las páginas)

public function Footer()

// Funcion que genera el output y su formato. Por ejemplo, si debe ser descargable como archivo,

// el nombre del archivo, etc.

public function Output($name = 'doc.pdf', $dest = 'I')

Si uno quisiera agregar un nuevo tipo de documento se deben seguir los siguientes pasos:

1- Crear clase generadora de documentos que extienda de TCPDF. Almacenar la clase en application/libraries/

2- Modificar el archivo application/models/documento y su función render(). Agregar una nueva condicion para inicializar el nuevo generador cuando el tipo de documento corresponda.

**Firma Digital Avanzada**

**Integración con HSM**

El objetivo de este capitulo es describir como Simple realiza Firma Digital Avanzada a través de un HSM y como se puede extender para utilizarla con otros proveedores de HSM.

HSM son las siglas de "Hardware Security Module" (Módulo de Seguridad Hardware). Un HSM es un dispositivo criptográfico basado en hardware que genera, almacena y protege claves criptográficas y suele aportar aceleración hardware para operaciones criptográficas.

Este mecanismo permite que Simple pueda realizar operaciones de firma digital al vuelo, en forma automática a través de un servidor externo (HSM) el cual es el encargado de realizar las firmas.

**Integración vía Webservice SOAP**

La implementación actual de Simple se integra con un HSM a través de un Webservice. Vía SOAP se envía el documento al servidor HSM, y este retorna firmado devuelta a Simple.

Esta es la descripción de la API SOAP:

**Metodo: IntercambiaDoc**

**Encabezado**

Parametro Valor

User Nombre de usuario en el HSM Password Contraseña del HSM TipoIntercambio 'pdf' o 'xml'

NombreConfiguracion Identificador de la configuración con la que se quiere firmar en el HSM. Ej: Varios firmantes en el HSM, cada uno con una configuración distinta

FormatoDocumento 'b64' que corresponde a Base64

**Parametro**

Parametro Valor

Documento Documento codificado en Base64

NombreDocumento Nombre del documento. Generalmente es el filename

Si el proveedor del HSM genera un webservice SOAP para el HSM siguiendo este esquema, se puede integrar directamente en Simple sin ningún desarrollo adicional.

Solamente bastaría con editar el archivo:

application/config/config.php

y setear los parámetros:

/\*

\* Configuracion del servidor HSM

\*/

$config['hsm\_url'] = '' // URL del endpoint del webservice HSM

$config['hsm\_user'] = '' // Nombre de usuario

$config['hsm\_password'] = '' // Contraseña

Luego de eso, automáticamente quedara habilitado en el editor de documentos de

Simple, un checkbox con la opción: "Firmar documento con HSM"

**Integración vía otros métodos**

Si se desea integrar con un HSM vía otro método, se debera reimplementar ciertas secciones de Simple. En particular, se debe editar el archivo:

application/models/documento.php

Dentro del método render($identifier, $key, $etapa\_id=null ,$filename, $copia) se encuentra el código que hace la llamada al HSM para realizar la firma.

Estas son las líneas encargadas de ese llamado:

if (!$copia && $this->hsm\_configuracion\_id) {

$client = new SoapClient($CI->config->item('hsm\_url'));

$result = $client->IntercambiaDoc(array(

'Encabezado' => array(

'User' => $CI->config->item('hsm\_user'),

'Password' => $CI->config->item('hsm\_password'),

'TipoIntercambio' => 'pdf',

'NombreConfiguracion' => $this->HsmConfiguracion->nombre,

'FormatoDocumento' => 'b64'

$filename)),

),

'Parametro' => array(

'Documento' => base64\_encode(file\_get\_contents($uploadDirectory .

'NombreDocumento' => $filename

)

));

file\_put\_contents($uploadDirectory . $filename, base64\_decode($result-

>IntercambiaDocResult->Documento));

}

Este código se ejecuta solamente cuando no se esta generando una copia del documento (Las copias no van firmadas) y cuando Simple esta configurado para funcionar con un HSM (En el config.php)

El código interior se debe reemplazar por el código custom que se necesite para conectarse con el HSM.