Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de:

Ingeniero en Ciencias Informáticas

Herramienta web para la generación de datos en bases de datos PostgreSQL

Autor(es):

Yanet Gonzalez Dieguez

Eddy Roy Velázquez Parra

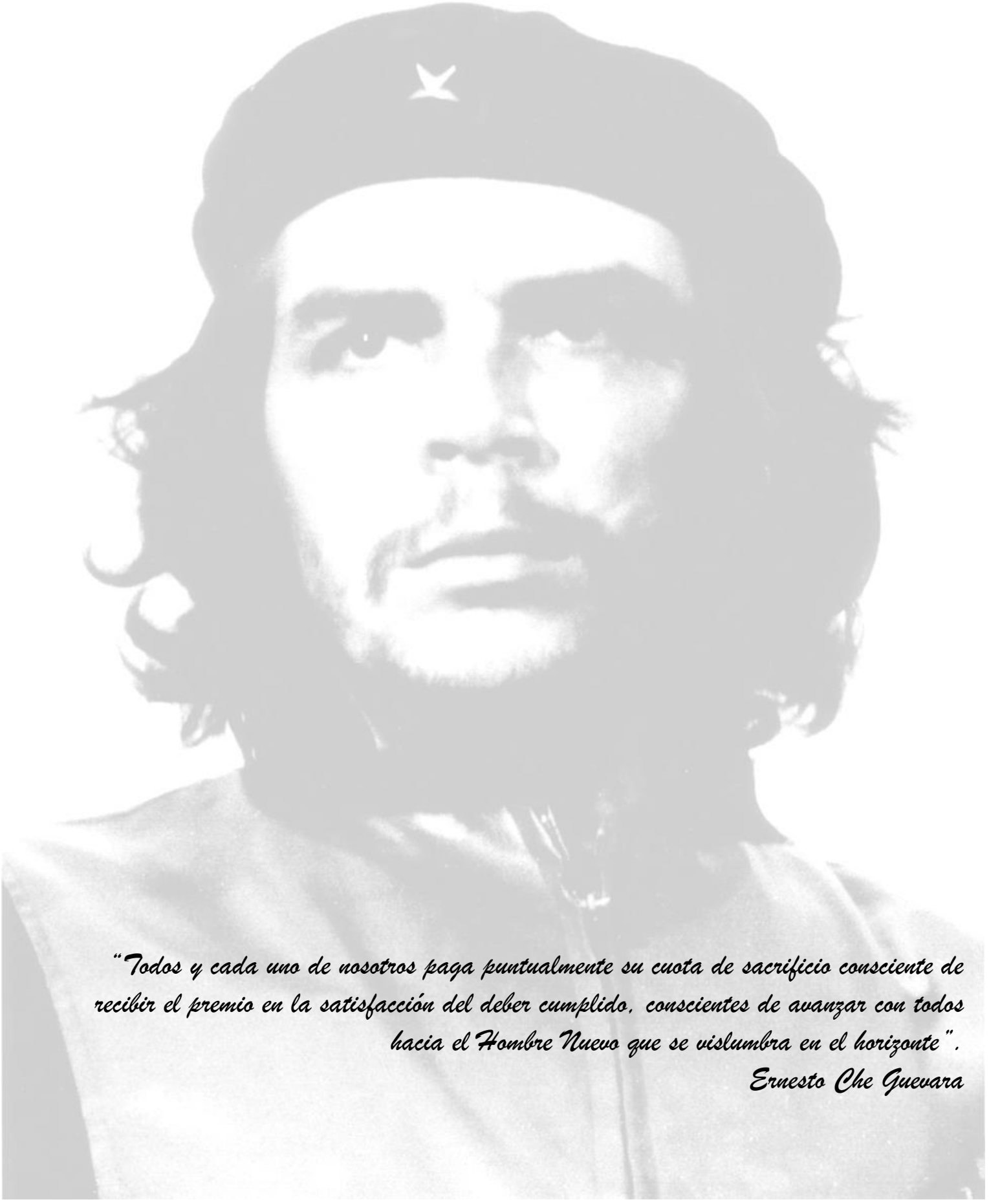
Tutor(es):

Ing. Glennis Tamayo Morales

Ing. Flavio Enrique Roche Rodríguez

La Habana, julio de 2015

"Año 57 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".

Ernesto Che Guevara

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Yanet Gonzalez Dieguez

Firma del autor

_____

Eddy Roy Velázquez Parra

Firma del autor

_____

Ing. Glennis Tamayo Morales

Firma del tutor

_____

Ing. Flavio Enrique Roche Rodríguez

Firma del tutor

Datos de contacto

Ing. Glennis Tamayo Morales

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Año de graduada: 2010

Años de experiencia: 4 años

Correo electrónico: gtamayo@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Ing. Flavio Enrique Roche Rodríguez

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Año de graduado: 2011

Años de experiencia: 3 años

Correo electrónico: feroche@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Agradecimientos

Yanet

Quiero empezar agradeciendo en primer lugar a mis padres que estuvieron siempre presentes en todos mis tropiezos, alegrías, tristezas y logros de la vida, por enseñarme a luchar para hacer realidad mis sueños, por todo su apoyo, sacrificio y dedicación. Por depositar en mi toda su confianza y demostrarme que a pesar de los obstáculos que se te presenten en la vida si se puede, por creer en mí, por todo el amor y el cariño infinito que siempre me han brindado. Por ayudar en mi formación como mujer, como persona y como profesional con los principios que siempre me inculcaron que con su ternura me transformaron en la mujer que soy hoy, los amo con todo mi corazón y no me va alcanzar la vida para seguir agradeciéndoles, pues para mí no cabe la menor duda que sin su ayuda y sus consejos no hubiera podido llegar hasta este momento. A mis hermanos Jose Ángel y Victor Manuel por ser los mejores hermanos del mundo, por ser cómplices de mi niñez, por ser tan importantes para mí, por los buenos momentos que hemos podido pasar juntos. Sólo mi corazón sabe cuán grande es el amor que siento por ellos, los amo. A mi tía Zoilita y mi prima Yumilka por brindarme tanto cariño desde mi niñez y estar siempre pendiente de cómo me iba en la universidad, les agradezco mucho. A toda mi familia en general los quiero y les doy las gracias, porque todos han puesto su granito de arena para que mi sueño se hiciera realidad.

A mis grandes amigas de Santiago Anizayli y Aliuska por sus consejos, por brindarme su apoyo y comprensión en momentos cuando más los necesitaba cuando me encontraba de pase. Las quiero mucho. A mis 3 tutores, porque para mí fueron en realidad 3 tutores los que tuve, por su dedicación. A Bonne, porque a pesar de que ya no se encuentre en la universidad supo guiarnos en un principio en el desarrollo del trabajo. A Flavio, muchas gracias por todo el tiempo que dedicó a apoyarnos en la realización de este trabajo. A Glennis gracias por guardarme un espacio siempre que la he necesitado a pesar de la carga de trabajo que siempre tenía, por su apoyo incondicional, por los buenos consejos, gracias y mil gracias. A mi dúo de tesis Eddy por su apoyo y comprensión en momentos cuando más lo necesitaba, por todas las horas

que puso en la realización de este trabajo, por el esfuerzo, dedicación, entrega y confianza, por los buenos y malos momentos que pasamos juntos. A todos los profesores que tuve durante la carrera que han contribuido en mi educación, en mi formación como profesional y como persona. A mi compañera de cuarto Aryanna por su apoyo y comprensión en momentos cuando más lo necesitaba, por ser una excelente amiga, a todas las amistades que conocí y sin dejar de mencionar a mis compañeros de grupo los actuales y los viejos.

Eddy Roy

Agradezco de manera especial a mis padres, por el apoyo, el esfuerzo y el sacrificio que supieron darme cuando más lo necesitaba, gracias a mi hermano por todo el cariño y amor que me ha sabido entregar. Gracias a mis abuelos por brindarme las fuerzas y el ánimo necesario para lograr esta meta. Agradezco a mis tíos, mis primos y toda mi familia que siempre estuvieron presente en el transcurso de estos cinco años. Gracias a Alexis por su confianza y su amistad incondicional no solo hacia mi familia sino hacia mí también, doy gracia a mi amigo Adrián, que más que amigo ha sido como un hermano. Agradezco de manera especial a mi dúo de tesis Yanet por su aguante, dedicación, cariño y confianza durante el desarrollo de este trabajo. Agradezco a mis compañeros de grupo los actuales y los viejos, a las amistades que he hecho en el transcurso de mis años en la UCI, a mis profesores que me formaron, enseñaron y educaron. También doy gracias a mis tutores y todas las personas que de una forma u otra hicieron posible la realización de este sueño.

Dedicatoria

Yanet

Este gran triunfo está dedicado a mis padres, a quienes les debo todo lo que tengo y lo que soy en esta vida, porque sin ellos nada de esto hubiera sido posible, por ser lo más grande que tengo en el mundo y haberme apoyado mucho durante toda la carrera. Les dedico este trabajo ante todo, porque no me han dejado claudicar en el logro de mis sueños. Son mi ejemplo a seguir, mis guías, especialmente por su amor, dedicación, comprensión y ayudarme a lograr este sueño de poder regalarles esta satisfacción de poder verme graduada.

Eddy Roy

Dedico el trabajo de diploma a mis padres Yuditza Parra y Eddy Velázquez, a mi hermano Pedro Luis, a mi abuelo Roy y a mis más cercanos amigos por ayudarme a cumplir mi sueño de convertirme en un profesional dedicado y regalarles la satisfacción de verme graduado.

Resumen

A raíz del avance tecnológico existente actualmente se han desarrollado productos informáticos que trabajan con diferentes gestores de bases de datos. Para que estos productos sean liberados deben pasar por un proceso de pruebas, permitiendo controlar que los datos almacenados en las bases de datos coincidan con los que realmente deben manejar. Para la realización de estas, los desarrolladores recurren a herramientas generadoras de datos, pues obtener la información suficiente para poblar bases de datos de forma manual sería una tarea realmente agotadora. Además, las aplicaciones ya existentes no tienen en cuenta restricciones tales como: formatos, longitudes, rangos y frecuencias de apariciones de valores de la información generada. El presente trabajo tiene como objetivo el desarrollo de una herramienta web para la generación de datos en bases de datos PostgreSQL. Para ello se realiza un estudio de varios generadores de datos existentes; así como las herramientas y tecnologías empleadas durante el desarrollo. Más adelante se realiza el análisis identificando las funcionalidades y quedando diseñada la solución propuesta. Del proceso de implementación se obtuvo como resultado una herramienta que permite la generación de datos en bases de datos PostgreSQL, dando solución al problema en cuestión, poblar bases de datos. Para validar el correcto funcionamiento de la aplicación se realizaron pruebas funcionales y de aceptación, arrojando como resultado diversas no conformidades que fueron resueltas. El desarrollo de la herramienta constituye una alternativa de gran ayuda en la realización de pruebas a los productos desarrollados en la Universidad de las Ciencias Informáticas (UCI).

Palabras claves: Bases de datos, generación de datos, herramienta.

Abstract

As a result of the existing technological advance today it had been developed software products that work with different database managers. For these products to be released they must go through a testing process, allowing to control that data stored in the database match those that should really be handled. For its creation, developers turn to data-generating tools, since obtaining sufficient information to populate databases manually would be a really exhausting task. Besides, existing applications do not consider restrictions such as formats, lengths, ranges and frequency of occurrences of values of the information generated. This present work have as objetive a developing a web tool for generating data in PostgreSQL databases. As for that it is made a study of different existing data generators, as well as technologies and tools used during development. Further analysis is done identifying functionalities being designed the proposed solution. From the implementation process it was obtained as a result a tool for generating data in PostgreSQL databases, giving solution to the problem a such, populate databases. To validate the proper functioning of the application functional and acceptance tests were performed, resulting in various non-conformities that were solved. The development of the tool is an alternative of a great help in testing the products developed at the University of Informatics Sciences (UCI).

Keywords: Databases, data generation, tool.

Índice

Índice de figuras

Índice de tablas

Introducción

El avance tecnológico que existe actualmente en el mundo se encuentra enmarcado fundamentalmente en el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), resaltando en especial el papel que desempeña la rama de la informática en este contexto. Con el propósito de brindar un acceso centralizado a grandes volúmenes de información en formato digital, se emplean las bases de datos.

"Una base de datos es una colección de datos organizados y estructurados según un determinado modelo de información que refleja no sólo los datos en sí mismos, sino también las relaciones entre ellos. Una base de datos se diseña con un propósito específico y debe ser organizada con una lógica coherente. Los datos podrán ser compartidos por distintos usuarios y aplicaciones, pero deben conservar su integridad y seguridad al margen de las interacciones de ambos". (Lapuente, 2013)

A partir del trabajo con las bases de datos surgen los Sistemas Gestores de Bases de Datos (SGBD), según Sara Álvarez, (Álvarez, 2007): *"Un SGBD se define como el conjunto de programas que administran y gestionan la información contenida en las bases de datos. Ayuda a definir los datos, mantener la integridad de los datos dentro de la bases de datos, manipular los datos, controlar la seguridad y privacidad de los datos".*

La creación de los SGBD relacionales supone un avance importante para facilitar la programación de aplicaciones con bases de datos, conseguir que los programas sean independientes de los aspectos físicos de estas y que en la actualidad sean mucho más seguras, flexibles y manejables en la mayoría de los casos. En las bases de datos relacionales, poblarlas corresponde a la creación de las tuplas que componen las tablas, donde las tuplas corresponden a los datos insertados en estas.

Diversas son las herramientas que han surgido en los últimos años para la generación de datos, pues se crean con el objetivo de minimizar tiempo y recursos en el proceso de prueba a las bases de datos. Evitan el trabajo difícil de insertar manualmente los datos en las tablas donde se introducen valores que en la aplicación final pueden estar validados de forma diferente, provocando errores a la hora de consultar los datos. Ejemplos que pueden ser útiles a la hora del llenado de las bases de datos son las que a continuación se mencionan: GenerateData, DBMonster y DataGenerator.

GenerateData es una herramienta libre que permite generar rápidamente grandes volúmenes de datos personalizados en una gran variedad de formatos para su uso en pruebas de software y rellenar bases de datos. DBMonster es una herramienta que genera datos de prueba al azar, proporcionando una interfaz fácil de usar. Datagenerator es una biblioteca y tiene una GUI (por sus siglas en inglés *Graphical User Interface*) para la generación de datos de prueba basados en reglas para diferentes SGBD como: MySQL, Firebird, Interbase, MSSQL, Oracle, SQLite y PostgreSQL.

Cuando se generan datos mediante herramientas web, estas brindan a miles de usuario la posibilidad de utilizar un servicio para poblar bases de datos. Dicho servicio puede ser usado a través de cualquier sistema operativo, ya que sólo necesita de un navegador para poder ser consumido, lo que aumenta su disponibilidad con respecto a una aplicación de escritorio con la misma funcionalidad.

De manera general las herramientas generadoras de datos existentes presentan una gran deficiencia, pues la mayoría no toman en cuenta las restricciones del negocio modelado que pueden ser: frecuencias de apariciones de valores, formatos, longitudes y rangos. Otras presentan una gran limitante, y es que son privativas, lo que es necesario pagar para poder obtenerlas y usarlas. Poblar bases de datos relacionales donde todos los datos insertados cumplan con todas las restricciones tanto del modelado relacional como las inherentes al negocio que se modela, constituye un gran reto. Actualmente no se cuenta con una herramienta capaz de resolver dichas restricciones que se manifiestan cuando se generan datos.

Por todo lo anteriormente expuesto, se define como **problema de investigación:** ¿cómo contribuir al proceso de población de las bases de datos en PostgreSQL?

Partiendo del problema planteado el **objeto de estudio** se enmarca en el proceso de poblar bases de datos relacionales, definiéndose como **campo de acción** el proceso de poblar bases de datos desarrolladas en PostgreSQL.

Para dar solución al problema planteado, se define como **objetivo general:** desarrollar una herramienta web para la generación de datos en bases de datos PostgreSQL que permita la población de bases de datos relacionales.

En correspondencia con el objetivo general, se plantean como **objetivos específicos:**

- Definir el marco teórico de la investigación para el desarrollo de la herramienta generadora de datos en bases de datos PostgreSQL.
- Realizar el análisis y diseño de la herramienta generadora de datos en bases de datos PostgreSQL.
- Implementar la herramienta generadora de datos en bases de datos PostgreSQL.
- Validar la herramienta implementada para la generación de datos en bases de datos PostgreSQL.

Para dar cumplimiento a los objetivos específicos anteriormente planteados, se proponen las siguientes **tareas de investigación:**

- Revisión de la bibliografía para la construcción del marco teórico de la investigación.
- Caracterización de las soluciones informáticas de generación de datos para recopilar las principales características e incluirlas en la herramienta a desarrollar.
- Selección de la metodología, herramientas y tecnologías a utilizar para el desarrollo de la herramienta generadora de datos.
- Identificación y descripción de las funcionalidades básicas de la herramienta generadora de datos para guiar el proceso de implementación.
- Implementación de las funcionalidades identificadas para la generación de datos en bases de datos PostgreSQL.
- Diseño de los casos de prueba para la realización de pruebas al generador de datos.
- Aplicación de pruebas de software para validar que el generador de datos responde a las necesidades del usuario final.

Para dar cumplimiento a las tareas planteadas, realizando la búsqueda y procesamiento de la información se utilizan los siguientes **métodos de investigación:**

Como **métodos teóricos** se utilizan:

- **Histórico-Lógico:** permitió realizar un estudio para comprender el proceso de generación de datos y las tendencias actuales de las herramientas que realizan esa operación.
- **Análisis-Síntesis:** permitió analizar los documentos y teorías relacionados con la generación de datos, posibilitando a partir de la cantidad de información necesaria y suficiente, resumir dicha información en busca de los objetivos perseguidos con la investigación, logrando arribar a conclusiones.

- **Inductivo-Deductivo:** permitió que a partir de conocimientos generales se pudieran derivar casos particulares, luego de haber realizado un razonamiento lógico a partir de la información recopilada.
- **Modelación**: se utiliza para la realización de los modelos que intervienen en el desarrollo de la herramienta de acuerdo a las características específicas de la aplicación.

Como **métodos empíricos** se utilizan:

- **Análisis documental:** es utilizado para analizar todos los documentos existentes y conceptos relacionados al tema que sustentan la investigación y así poder seleccionar los más apropiados para el desarrollo de la investigación.
- **Entrevista:** se utiliza para recopilar la información tanto primaria como secundaria que tributa al desarrollo de la herramienta.

Las siguientes **preguntas científicas** guiarán el desarrollo de la investigación:

- ¿Cuáles son los referentes teóricos relacionados con la implementación de un generador de datos?
- ¿Cuáles son las principales características que debe tener el generador de datos que permita la población de las bases de datos en PostgreSQL?
- ¿Cómo desarrollar el generador de datos para la población de las bases de datos en PostgreSQL?
- ¿Cómo validar el correcto funcionamiento del generador de datos para la población de las bases de datos en PostgreSQL?

El presente trabajo se encuentra estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica

En el presente capítulo se realiza un estudio sobre las herramientas generadoras de datos existentes de forma general con funcionalidades similares a la solución propuesta. Se menciona la metodología, herramientas y lenguajes utilizados como soporte al proceso de desarrollo de *software* que más se ajusta a las necesidades del trabajo. Se realiza una selección y descripción de las tecnologías necesarias para el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL.

Capítulo 2: Análisis y Diseño

El presente capítulo tiene como principal finalidad abordar contenidos relacionados con el análisis y diseño

del sistema. Se realiza una descripción detallada de la propuesta de solución para el problema de investigación identificado. Se generan los artefactos que propone la metodología de desarrollo de *software* XP para el diseño de la aplicación. Además se definen los patrones de diseño y la arquitectura que se utilizará para el desarrollo de la solución propuesta.

Capítulo 3: Implementación y Prueba

El contenido que se aborda en el capítulo está relacionado con la descripción de la implementación del sistema para darle solución a las historias de usuario definidas en el capítulo anterior. Se realizan las tareas de ingeniería, se definen los estándares de codificación y se especifican las pruebas a las que fue sometida la aplicación para validar que cumple con las expectativas del usuario final.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se realiza un estudio sobre las herramientas generadoras de datos existentes de forma general con funcionalidades similares a la solución propuesta. Se menciona la metodología, herramientas y lenguajes utilizados como soporte al proceso de desarrollo de *software* que más se ajusta a las necesidades del trabajo. Se realiza una selección y descripción de las tecnologías necesarias para el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL.

1.1. Generación de datos

El generador de datos de las bases de datos surge de la necesidad de llenar sus campos de forma correcta, simplificando el trabajo de los desarrolladores a la hora del llenado y por ende disminuir el tiempo de prueba del software. La generación de datos es de vital importancia para los desarrolladores de bases de datos cuando necesitan poblarlas con un gran volumen de datos que de otra forma llevarían consigo un gasto considerable de tiempo y recursos.

1.1.1. Herramientas de generación de datos

Debido al progreso tecnológico alcanzado en la actualidad, varias son las herramientas que han surgido para la generación de datos y que apoyan todo el proceso de obtención de datos que van desde herramientas *desktop* hasta *web*, algunas propietarias y otras libres. Las herramientas de generación de datos son aplicaciones que generan datos de prueba para las diferentes tablas de las bases de datos. Con su surgimiento ha aumentado en gran medida la productividad por parte de los desarrolladores, brindando un grupo de funcionalidades que facilitan el manejo de los datos.

A continuación se presenta un estudio sobre las principales características que presentan algunas de las herramientas de generación de datos que existen. Teniendo en cuenta en dicho estudio, no sólo el SGBD PostgreSQL, sino también otros, con el objetivo de realizar una investigación más profunda acerca de las funcionalidades básicas que debe cumplir un generador de datos. Dicha investigación posibilita que sea la base fundamental para seleccionar las características más comunes que van a describir la herramienta *web* a desarrollar. Para el estudio se seleccionaron 12 herramientas de generación de datos, de ellas siete

son propietarias: ForSQL Data Generator 1.0, Data Generator para SQL Server 2.2, Data Generator para DB2, Data Generator para Oracle 2.3, Data Generator para MySQL 2.3, Data Generator para PostgreSQL 2.3 y Generador de datos para PostgreSQL Datanamic; y cinco libres: GenerateData, DBMonster, Datagenerator, Plugin de generación de datos para la Herramienta de Administración de Bases de Datos HABD y Generador de datos ficticios en bases de datos PostgreSQL y MySQL.

ForSQL Data Generator 1.0

Es un generador de datos automático para pruebas de bases de datos a gran escala. Ahorra innumerables horas y hasta días que serían gastados en la entrada y configuración de los datos a mano. Brinda la posibilidad de trabajar de forma sencilla y es utilizado para varios gestores de bases de datos. Otro aspecto a tener en cuenta es su capacidad para llenar espacios usando varios métodos diferentes que se extienden desde valores arbitrarios hasta el relleno con verdaderos datos de plantillas.

Data Generator para SQL Server 2.2

Tiene gran utilidad para la generación de datos de prueba en varias tablas de bases de datos Microsoft SQL a la vez. La aplicación asistente permite definir tablas, campos para la generación, establecer rangos de valores y obtener listas de valores de las consultas *sql* de forma sencilla. Proporciona una aplicación de consola que permite la generación mediante el uso de plantillas. Brinda una amplia variedad de parámetros de generación para cada tipo de dato en su respectivo campo. Tiene la capacidad para establecer los valores nulos para cierto porcentaje de datos de carácter obligatorio.

Data Generator para DB2

Es una herramienta para la generación de datos de prueba para las tablas de bases de datos DB2[1] con la posibilidad de guardar y editar los *script*. Permite el establecimiento de rangos de valores, definición de tablas, campos y listas de valores de forma manual. Su utilidad se basa en que puede ayudar a simular el entorno de producción de las bases de datos. Permite mediante líneas de comando usar plantillas de generación. Presenta soporte *unicode* (código único) y una interfaz sencilla. Brinda la posibilidad de guardar y editar los datos generados en el *script sql* sin ejecutar consultas en el servidor.

[1] DB2: es una marca comercial, propiedad de IBM (empresa multinacional estadounidense de tecnología y consultoría), bajo la cual se comercializa un sistema de gestión de base de datos.

Data Generator para Oracle 2.3

Es una herramienta eficaz que ayuda a generar datos de prueba a varias tablas de bases de datos de Oracle a la vez. La aplicación asistente permite definir tablas, campos para la generación, establecer rangos de valores y obtener listas de valores a partir de consultas *sql*. Proporciona una aplicación de consola que permite la generación mediante el uso de plantillas. Su interfaz de asistente es fácil de usar y presenta soporte *unicode*. Brinda la posibilidad de guardar y editar los datos generados en el *script sql* sin ejecutar consultas en el servidor. Permite generar para diferentes esquemas de bases de datos. Tiene la capacidad para generar una vista previa de los datos. Brinda una amplia variedad de generación de parámetros para cada tipo de campo y la capacidad de establecer los valores nulos para cierto porcentaje de datos de carácter obligatorio.

Data Generator para MySQL 2.3

Tiene gran utilidad para la generación de datos de prueba a varias tablas de bases de datos MySQL a la vez. Permite definir tablas, campos, rangos de valores establecidos y la generación mediante el uso de plantillas por defecto. Su interfaz de asistente es fácil de usar. Genera datos para diferentes bases de datos en un equipo remoto. Utiliza los resultados de consultas *sql* como lista de valores para la generación y una amplia variedad de parámetros a poblar para cada tipo de campo, que son guardados para la generación. Además utiliza líneas de comandos para generar datos utilizando el archivo de plantilla.

GenerateData

Es un proyecto de *open source* (código abierto) que permite generar grandes volúmenes de datos personalizados en una gran variedad de formatos para su uso en pruebas de *software*. Ofrece tanto una *demo online* (demostración en línea) donde probar sus características como la posibilidad de descargar el de utilizar varios tipos de datos: nombres, números de teléfono, direcciones de correo electrónico, ciudades, estados, provincias, condados, fechas, direcciones, rangos de números, cadenas alfanuméricas y textos. La versión gratuita permite generar hasta 200 registros de una vez.

DBMonster

Es una herramienta que genera datos de prueba al azar y los datos son colocados en las bases de datos *sql*, proporcionando una interfaz amigable. Prueba el rendimiento de las aplicaciones de estas bajo carga pesada. No admite modelo avanzado de generación de datos de más de una tabla al mismo tiempo.

Datagenerator

Es una herramienta para rellenar tablas de carga con datos semialeatorios. Utiliza varias bibliotecas y tiene una GUI para la generación de datos de prueba basados en reglas para diferentes bases de datos como: MySQL, Firebird, Interbase, MSSQL, Oracle, SQLite y PostgreSQL. Los usuarios pueden especificar secuencias, textos, números aleatorios, columnas y los archivos como fuentes para las tablas.

1.1.2. Herramientas de generación de datos para PostgreSQL

Data Generator para PostgreSQL 2.3

Es una herramienta para la generación de datos de prueba a las tablas de las bases de datos en PostgreSQL, con la posibilidad de guardar y editar secuencias de comandos. Su utilidad se basa en ayudar a simular el entorno de producción de las bases de datos y el relleno de varias tablas con datos de prueba. Al mismo tiempo definir tablas y campos para la generación de datos, rangos de valores establecidos, definir listas de valores de forma manual o seleccionarlos de consultas *sql*. Tiene parámetros de generación establecidos para cada tipo de campo. Proporciona una aplicación de consola que permite generar datos de prueba de PostgreSQL mediante el uso de plantillas de generación. Brinda la posibilidad de generar datos de prueba a varias tablas y su uso permite cargar los valores para los campos de archivo.

Generador de datos para PostgreSQL Datanamic

La herramienta presenta características que ayudarán a generar datos de prueba significativos para bases de datos PostgreSQL con rapidez. Permite poblarlas directamente o generar *script* de inserción. Personaliza los generadores de datos de la forma que desee. Tiene una gran colección de idiomas, países predefinidos y crea datos definidos por los usuarios. Tiene limitaciones de 30 días de prueba y la generación máxima de 50 registros por tabla en cada procesamiento.

Generador de datos ficticios en bases de datos PostgreSQL y MySQL

Es una herramienta de escritorio que permite generar datos para las bases de datos PostgreSQL y MySQL. Por defecto brinda la posibilidad de generar datos de diferentes formas: aleatoria, incremental, de otra tabla y de un archivo externo. Ofrece una interfaz de asistente fácil de usar y tiene la capacidad de generar datos en un *script sql*. Permite generar datos ficticios de forma aleatoria para PostgreSQL generando los siguientes tipos de datos: *integer, double, text, char, date* y *float*. Para dicho tipo de

generación el usuario podrá trabajar con dos interfaces, una común para los tipos de datos: *integer*, *double*, *date* y *float*; y otra para los tipos de datos: *char* y *text*.

Plugin de generación de datos para la Herramienta de Administración de Bases de Datos (HABD)

HABD tiene como característica fundamental que presenta una arquitectura basada en *plugin*, pues facilita que los desarrolladores puedan incorporar nuevos servicios de forma sencilla por la flexibilidad que permite. Brinda la posibilidad de generar campos mostrando un listado con el nombre y el tipo de dato de los atributos que contiene la tabla que será generada. Permite generar datos aleatorios especificando los parámetros necesarios para cada atributo en dependencia del tipo de dato que tenga. Al realizar el proceso de generación de datos los cambios se guardarán en las bases de datos a la cual se está conectado, permitiendo además generar ayuda y datos desde una lista. Especifica como parámetro la cantidad de datos que se desean generar para cada tabla y para qué atributos se generarán los datos. En cada uno de los atributos se especificará el tipo de generación de datos a utilizar, ya sea de forma aleatoria o desde una lista.

Luego de un estudio realizado acerca de las diferentes herramientas de generación de datos que existen, para el análisis de las principales características de los generadores de datos, se selecciona 1 herramienta libre y 2 propietarias. Se elige dentro de las libres estudiadas a GenerateData, porque permite generar grandes volúmenes de datos, brindando una interfaz *web*. No se seleccionó DBMonster ni Datagenerator, puesto que a pesar de ser libres no son utilizadas frecuentemente, imposibilitando la consulta de bibliografías actualizadas referentes a dichas herramientas. Como tampoco se escogió Generador de datos ficticios en bases de datos PostgreSQL y MySQL, y Plugin de generación de datos para la Herramienta de Administración de Bases de Datos (HABD), debido a que no cumplen con un grupo de condiciones a la hora de generar datos como las frecuencias de apariciones de valores, formatos y rangos. Teniendo en cuenta que la HABD necesita de un *plugin* para ser utilizada en el puesto de trabajo; sin embargo, la herramienta *web* para la generación de datos en bases de datos PostgreSQL con sólo un navegador es suficiente para hacer uso de ella.

Dentro de las propietarias se selecciona Data Generator para Oracle 2.3, debido a que permite guardar y editar los datos que se generan en el *script sql* sin tener que ejecutar consultas en el servidor, ofreciendo un conjunto de funcionalidades que de forma sencilla permiten a los usuarios crear bases de datos con un conjunto de restricciones acorde a sus necesidades. Se seleccionó también como herramienta propietaria

Data Generator para PostgreSQL 2.3, debido a que genera varios tipos de datos propios del gestor PostgreSQL, siendo el gestor sobre el cual se desarrollará la aplicación, permitiendo utilizarla como apoyo en la visualización de los parámetros de los datos en el sistema. Las demás herramientas privativas no se tuvieron en cuenta, debido a que presentan deficiencia en cuanto a características esenciales que deben cumplir los modelos generados, por ejemplo, la falta de ayuda por parte de dichas herramientas para simular el entorno de producción de las bases de datos, imposibilitando así la calidad del llenado de varias tablas con datos de prueba. A continuación se evidencian las características fundamentales que debe tener un generador de datos, realizándose una comparación con los generadores de datos escogidos para el estudio y poder seleccionar las características que va a cumplir la herramienta para la generación de datos en bases de datos PostgreSQL.

Tabla 1. Características principales de las herramientas de generación de datos.

Características de las herramientas	**GenerateData**	**Data Generator para Oracle 2.3**	**Data Generator para PostgreSQL 2.3**
Uso de plantillas de generación.	No	Si	Si
Soporte de código único.	No	Si	No
Guardar secuencias de comandos.	No	No	Si
Definición de tablas.	Si	Si	Si
Código abierto.	Si	No	No
Editar secuencias de comandos.	No	No	Si
Generación de grandes volúmenes de datos.	Si	No	No
Generación de vista previa de los datos.	No	Si	No
Definición de rangos de valores.	Si	Si	Si
Carga de archivos para los campos con valores.	No	No	Si
Generación de datos en varios esquemas.	No	Si	No
Utilización de varios tipos de datos.	Si	Si	Si
Interfaz *web*.	Si	No	No
Simulación del entorno de producción.	No	No	Si
Definición de campos.	Si	Si	Si

Definición de valores de forma manual.	No	No	Si
Establecimiento de valores nulo para datos de carácter obligatorio.	No	Si	No
Frecuencias de apariciones de valores.	No	No	No
Formatos.	No	No	No
Longitudes.	No	No	No
Rangos.	No	Si	Si

Partiendo de las revisiones bibliográficas e investigaciones realizadas sobre los generadores de datos antes mencionados se proponen las siguientes características que incluirá el generador de datos a desarrollar en la presente investigación.

- Uso de plantillas de generación.
- Definición de tablas.
- Definición de rangos de valores.
- Utilización de varios tipos de datos.
- Definición de campos.
- Código abierto.
- Interfaz *web*.
- Definición de valores de forma manual.

Luego del estudio realizado se propone el desarrollo de una nueva herramienta de generación de datos, debido a que las estudiadas no cumplen con las características básicas que se necesitan. Fundamentalmente la mayoría no toman en cuenta las restricciones del negocio modelado que pueden ser: frecuencias de apariciones de valores, formatos, longitudes y rangos.

El desarrollo de la herramienta mediante el uso de aplicaciones *web,* permite la disponibilidad del sistema a través de dispositivos que tengan un navegador *web* como: ordenadores, teléfonos móviles, *tablet*, entre otros. Brindando además la posibilidad de actualizar la aplicación en el servidor, de tal modo que todos los usuarios puedan utilizarla en el momento, no se obliga al usuario a utilizar un sistema operativo obligatorio y no hay problemas de incompatibilidad entre versiones, porque todos trabajan con la misma aplicación en el servidor. Por todo lo anteriormente planteado son creadas las bases para desarrollar una herramienta *web* para la generación de datos en bases de datos PostgreSQL y no una de *desktop*.

1.2. Motor de generación datafiller

El *datafiller* es un *script* que permite generar datos de manera aleatoria para poblar bases de datos en PostgreSQL y MySQL, con el objetivo de darles valores a los datos. Además de generar datos aleatorios presenta características, tales como:

- Generador de datos que cubren los diferentes tipos de datos y sus posibles combinaciones.
- Valores por defecto de esos datos en base a las restricciones de clave y tipo.
- Configuración mínima en el tamaño de las tablas utilizando sus opciones.
- Generar datos a partir de archivos externos con la descripción y el formato adecuado.
- Es de código abierto.
- Los datos se caracterizan por tener una longitud y un uso en dependencia de su representación en las tablas. (Coelho, 2014)

Partiendo de las necesidades del cliente, el motor de generación a emplear fue modificado por un grupo de desarrolladores del Centro de Tecnologías de Gestión de Datos (DATEC), permitiendo soportar más parámetros de acuerdo a los tipos de datos que maneja. Se podrá utilizar durante la implementación para la creación del propio motor de generación de la herramienta. Teniendo en cuenta además que los datos a generar cumplirán con un formato adecuado y una longitud deseada en dependencia de los tipos de datos, contando con la opción de poder utilizar archivos externos con la información requerida para la generación.

1.3. Metodología de desarrollo de software

"La metodología de desarrollo de software se describe básicamente como el conjunto de herramientas, técnicas, procedimientos y soporte documental para el desarrollo de sistemas de información. Se dividen en dos grandes grupos, las pesadas o tradicionales y las blandas o ágiles. Las metodologías tradicionales se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar y requiere de una extensa documentación, mientras las ágiles no son más que la gestión adaptativa que permite llevar a cabo proyectos de software, adaptándose a los cambios y evolucionando en forma conjunta con el software". (Rubio, 2014)

La metodología es un proceso que sirve de ayuda a los integrantes de un proyecto para realizar con mayor calidad el *software* a automatizar, es decir, un conjunto de métodos que se emplean para el desarrollo de sistemas automatizados. Es de vital importancia, porque a través de una metodología se podrá llegar al objetivo real del cliente. El proceso de desarrollo de *software* no es una tarea fácil, se debe contar con un proceso bien detallado y se necesita aplicar una metodología que sea capaz de llevar a cabo el control total del producto.

1.3.1. Extreme Programming (XP)

"Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un clima agradable de trabajo". (ingenieriadesoftware.mex, 2015)

XP es una metodología ágil que se basa en una comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. El cliente tiene derecho de quitar o cambiar los requisitos de acuerdo a su interés y el desarrollador decide cómo implementar los procesos, aclarar dudas con el usuario, estimar esfuerzos y cambiar requisitos según nuevos descubrimientos. Presenta una frecuente interacción entre el equipo de programación y el cliente, simplicidad en el código, corrección y una programación por parejas.

1.3.2. Scrum

"Scrum es una metodología ágil de gestión de proyectos de desarrollo de software, basada en un proceso de trabajo constante, iterativo e incremental". (Bahit, 2011)

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la forma de trabajar de equipos altamente productivos. Es una metodología de desarrollo muy simple que requiere un duro trabajo, porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Se emplea en entornos que trabajan con requisitos inestables y que requieren flexibilidad, es decir, está indicada para proyectos con un rápido cambio de requisitos.

1.3.3. Open Up

"Open Up es una metodología de desarrollo de software de código abierto, diseñado para equipos organizados, tomando una aproximación ágil[2] del desarrollo. Además es iterativa, mínima, completa y extensible". (Fernandes Agrela et al., 2009)

Es una metodología apropiada para proyectos pequeños y de bajos recursos, permitiendo disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito. Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas. Además permite detectar errores tempranos a través de un ciclo iterativo.

Se decide utilizar XP como metodología de desarrollo de *software*, debido a que el equipo de desarrollo viene trabajando con ella desde años anteriores, facilitando su comprensión acerca de los artefactos que genera. Gran parte del éxito de su utilización se debe a que es el cliente quien conduce constantemente el trabajo, aportándole mayor valor del negocio y los programadores puedan resolver de forma inmediata cualquier duda asociada. Además centra la atención del equipo en desarrollar primero las funcionalidades y no en una profunda documentación del proceso de desarrollo. Para su empleo se definieron 5 meses de trabajo, optimizando de esta forma el tiempo de desarrollo. Permite realizar el sistema en parejas para complementar los conocimientos; presentando un código sencillo, además de la poca documentación a elaborar para el desarrollo de la aplicación.

1.4. Lenguaje de modelado

"El Lenguaje Unificado de Modelado UML (por sus siglas en inglés Unified Modeling Language) es un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software. Es una herramienta propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por analistas funcionales y analistas programadores ". (Krall, 2015)

UML fue creado por la necesidad de utilizar un lenguaje estándar para el modelado de sistemas, debido a que existía una gran diversidad de modelos o métodos. Es el lenguaje de modelado de sistemas de

[2] Aproximación ágil: en open up es un proceso unificado que incorpora técnicas ágiles probadas.

software que permite visualizar, especificar, construir, detallar los artefactos en el sistema y documentar un sistema de software orientado a objetos. Es utilizado para entender, diseñar, examinar, configurar, mantener y controlar la información sobre los sistemas.

Se utiliza UML, puesto que es un lenguaje muy utilizado a nivel mundial y también en la UCI. Además el equipo de desarrollo presenta vasta experiencia en el uso del lenguaje, contribuyendo a obtener un diseño con alta calidad y ahorrando tiempo en el estudio de otros lenguajes de modelado. Brinda la posibilidad de entender a partir del diseño realizado el funcionamiento de la herramienta.

1.5. Herramienta de modelado

"Las herramientas de modelado de sistemas informáticos son herramientas que se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Permiten crear un simulacro del sistema, a bajo costo y riesgo mínimo. A bajo costo, porque es un conjunto de gráficos y textos que representan el sistema, pero no son el sistema físico real (el cual es más costoso). Además minimizan los riesgos, porque los cambios que se deban realizar (por errores o cambios en los requerimientos), se pueden realizar más fácil y rápidamente sobre el modelo que sobre el sistema ya implementado". (Alegsa, 2015)

Las herramientas de Ingeniería Asistida por Computación CASE (por sus siglas en inglés *Computer Aided Software Engineering*) tienen como principales objetivos aumentar la calidad en el desarrollo del *software* y mejorar su planificación. Facilitan el uso de las diferentes metodologías propias de la ingeniería del *software*. Las herramientas CASE presentan mínima redundancia, o sea, el aumento de redundancia disminuye la transparencia del modelo y aumenta las tareas de mantenimiento, ayudando en todos los aspectos del ciclo de vida de desarrollo del *software*.

1.5.1. Visual Paradigm 8.0

"Visual Paradigm para el Lenguaje de Modelado Unificado (VP-UML) es una herramienta UML. Está diseñada para cualquier persona que esté interesada en la construcción de sistemas de software fiable a gran escala con un enfoque orientado a objetos. VP-UML soporta los últimos estándares de la notación UML". (download.com, 2013)

Visual Paradigm al disponer de un repositorio común es posible visualizar el mismo elemento en varios diagramas evitando duplicidades. Permite integrarse con otras aplicaciones como herramientas ofimáticas aumentando la productividad. Genera código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación. Está diseñada para un amplio grupo de usuarios que se encuentren interesados en realizar *software* a grandes escalas. Los desarrolladores la utilizan para facilitar el modelado simultáneo, almacenar los archivos de proyectos y hacer un seguimiento de los cambios que realicen.

Se decidió utilizar Visual Paradigm 8.0, debido a que es una herramienta CASE profesional que soporta el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de diagramas. Presenta licencia gratuita, fácil de instalar y actualizar, por lo que todas las personas pueden trabajar con la herramienta. Además como principales características se destacan su robustez, usabilidad, portabilidad y brinda un ambiente de trabajo que facilita la visualización y manipulación del proyecto modelado.

1.6. Lenguaje de programación

"Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo". (Kioskea.net, 2015a)

A raíz del desarrollo de los lenguajes de programación, las tendencias y necesidades de las plataformas, se encuentran los lenguajes de programación para la *web*. Durante la evolución de esos lenguajes, se observa que han pasado de ser estáticos a conocerse como lenguajes de programación para la *web* dinámica. Sin confundir el término dinámico como la interacción del sistema con el usuario, sino como la interacción del usuario con datos a través del sistema. En la actualidad pueden clasificarse como lenguajes de programación de alto nivel y bajo nivel. Para el desarrollo de aplicaciones *web* se encuentra PHP como lenguaje de programación de alto nivel.

1.6.1. PHP 5.3

PHP (por sus siglas en inglés *Hypertext Preprocessor*) es un lenguaje *script* interpretado en el lado del servidor que es utilizado para la generación de páginas *web* embebidas en páginas HTML[3] (por sus siglas en inglés *HyperText Markup Language*) y ejecutadas en el servidor. Tiene capacidad de conexión para la mayoría de los sistemas generadores de datos. Es de código abierto, por lo que se presenta como una alternativa de fácil acceso para todos.

"Php 5.3 ofrece un amplio rango de nuevas funcionalidades, añadiendo soporte para espacios de nombres, enlaces estáticos, etiquetas y soporte nativo para funciones anónimas. Brinda nuevas sintaxis, formas de declaración, operadores ternarios que tienen una nueva forma abreviada y es posible el acceso dinámico a métodos estáticos". (PHP.net, 2015)

Se utiliza PHP 5.3 por ser un lenguaje de programación multiplataforma, diseñado originalmente para la creación de páginas *web* dinámicas. Estando en correspondencia además con el *framework* de desarrollo escogido Symfony, el cual utiliza en su versión 2.3 este propio lenguaje. Soporta las conexiones para el SGBD utilizado, PostgreSQL.

1.7. Entornos de desarrollo

"Un Entorno de Desarrollo Integrado IDE (por sus siglas en inglés Integrated Development Environment) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Puede denominarse como un entorno de programación que ha sido tratado como una aplicación. Esto significa que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica". (Editorbfb, 2011)

IDE es un programa que permite desarrollar código en un lenguaje determinado. Brinda la posibilidad de un espacio para la escritura de código con cierta ayuda en su generación e indicar los errores de sintaxis que se cometan por parte del programador, compilar, ejecutar el código escrito y organizar los proyectos. En algunos lenguajes un IDE puede funcionar como un sistema en tiempo de ejecución, donde se permite

[3] HTML: es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código para la definición de contenido de una página web como textos, imágenes, videos, entre otros.

utilizar el lenguaje de programación en forma interactiva sin necesidad de trabajo orientado a archivos de texto.

1.7.1. PhpStorm 8.0

"PhpStorm *es un IDE de programación desarrollado por JetBrains*[4]*. Es uno de los entornos de programación más completos de la actualidad. Permite editar código no sólo del lenguaje de programación PHP como lo indica su nombre, sino una gran variedad de lenguajes. Entre sus características principales se encuentra la gestión de proyectos fácilmente, proporciona autocompletado de código de manera sencilla y presenta una sintaxis abreviada".* (editoresdecodigo.com, 2014)

Se utiliza PhpStorm 8.0 al inicio de la implementación, debido a que permite el auto completamiento de código y comprobar errores al instante. Es utilizado en el trabajo con el *framework* de desarrollo de aplicaciones *web* Symfony. Al margen de todas las características habituales en los entornos de desarrollo, es el único que ofrece una integración casi perfecta con Symfony gracias a su *plugin* para Symfony 2. Algo que destaca es la ejecución del código en la misma interfaz del IDE; así como la interpretación y visualización inmediata del código PHP que se está desarrollando.

1.7.2. NetBeans 8.0

"NetBeans IDE es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Es un producto libre, gratuito sin restricciones de uso y está escrito en Java, pero puede servir para cualquier otro lenguaje de programación". (netbeans.org, 2015)

Se decidió utilizar NetBeans 8.0 durante el resto de la implementación, debido a la función que realiza el *debugger* (depurador) que permite encontrar donde están los errores en el código, o sea, visualizar paso a paso como funciona un proceso. Presenta un entorno sencillo, facilitando de esta manera su uso y por consiguiente un mejor resultado en el desarrollo de la herramienta. Funciona en diversas plataformas, permitiendo agilizar el desarrollo de grandes aplicaciones de varias índoles. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones *web*, control de versiones y sus funcionalidades son ampliables mediante la instalación de paquetes. Además resulta más cómodo su

[4] Es una empresa de desarrollo de software cuyas herramientas están dirigidas a los desarrolladores de software y administradores de proyectos.

trabajo, gracias al conocimiento que tiene el grupo de desarrolladores del IDE, ahorrando tiempo en el estudio de nuevos entornos de desarrollo.

1.8. Sistema gestor de bases de datos

1.8.1. PostgreSQL 9.3

"PostgreSQL es un sistema gestor de bases de datos objeto-relacional. Distribuido bajo licencia BSD (por sus siglas en inglés Berkeley Software Distribution) y con su código fuente disponible libremente. Es el SGBD de código abierto más potente del mercado y en sus últimas versiones se encuentra a la altura de otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Trabaja fácilmente con grandes cantidades de datos y posee una alta concurrencia de usuarios accediendo a la vez al sistema". (Martínez, 2010)

Se decidió utilizar PostgreSQL por ser el gestor sobre el cual se desarrollará la aplicación, debido a que es libre y cualquier usuario puede hacer uso de él, permitiendo asegurar la integridad de los datos. Además es el SGBD que se utiliza en el centro DATEC de la facultad 6 y que se está abogando a utilizar en todo el país para el trabajo con las bases de datos. Es altamente adaptable a las necesidades del cliente, puesto que tiene excelentes interfaces de instalación y administración. Presenta una documentación bien organizada y pública, manteniendo un gran avance en el desarrollo de las bases de datos. Es utilizada su versión 9.3, porque es la versión actual sobre la cual se empezará a realizar la herramienta.

1.8.2. Administrador de bases de datos. PgAdmin III 1.18.1

Un administrador de bases de datos gestiona y mantiene las bases de datos informatizadas. Se cercioran que sean seguras y estén actualizadas. Crean sistemas de respaldo para que los datos no se pierdan si surgen problemas en las bases de datos. Es importante para garantizar la seguridad de las bases de datos, porque son los responsables de la integridad y la disponibilidad de los datos. Permiten diseñar la distribución de los datos y las soluciones de almacenamiento.

"PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia de open source. Fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos

complejas". (debian.org, 2015)

Se decidió utilizar PgAdmin III, porque al ser libre es muy utilizado en el mundo y también en la UCI. Brinda una forma sencilla a la hora de interactuar con los datos almacenados en dicho administrador. Además su interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración de bases de datos para aplicaciones que utilicen dicho gestor de bases de datos.

1.9. Framework

Un *framework* es un marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a gran escala de componentes de *software* para el desarrollo rápido de aplicaciones. Desde el punto de vista del desarrollo de *software*, un *framework* es una estructura de soporte definida como plataforma, entorno o marco de trabajo, en la cual, otro proyecto de *software* puede ser organizado y desarrollado. Dentro de los objetivos principales que persiguen se encuentra acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Brinda la posibilidad de facilitar el desarrollo de *software* y evitar detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos.

"Un framework de aplicaciones web es un tipo de framework que permite el desarrollo de sitios web dinámicos, servicios web y aplicaciones web. El propósito de este tipo de framework es permitir a los desarrolladores construir aplicaciones web y centrarse en los aspectos interesantes, aliviando la típica tarea repetitiva asociada con patrones comunes de desarrollo web". (Alcalde, 2012)

La mayoría de los *frameworks* de aplicaciones *web* proporcionan tipos de funcionalidades básicas comunes, tales como, sistemas de *templates* (plantillas), manejo de sesiones de usuario y persistencia de datos. Normalmente promueven la reutilización y conectividad de los componentes; así como la reutilización de código y la implementación de bibliotecas para el acceso a las bases de datos.

1.9.1. Symfony 2.3.1

Symfony2 es una colección de más de 20 librerías independientes que se pueden utilizar dentro de cualquier proyecto PHP. Las librerías llamadas componentes de Symfony2 son útiles prácticamente para cualquier aplicación, independientemente de la manera en que se desarrolle el proyecto.

Es utilizado Symfony en su versión 2.3.1 como *framework* de desarrollo *web*, porque brinda la posibilidad de la reutilización de código y el uso de bibliotecas como soporte al desarrollo de la herramienta. Tiene la ventaja de cuando es creado un proyecto son creadas a la par un grupo de funcionalidades por defectos que permiten configurar de manera más sencilla las diferentes operaciones que se desean utilizar, de forma que organizan el trabajo para los desarrolladores. Presenta elegancia en el código, pues debido al uso de la jerarquía para su interpretación por el compilador obliga a la limpieza y organización del código a la hora de su programación.

Conclusiones del capítulo

En el capítulo se realizó un estudio acerca de las principales herramientas de generación de datos que existen, permitiendo identificar las características que contendrá el sistema a desarrollar; así como la utilización del *datafiller* como base para la creación del propio motor de generación de la herramienta. Se determinó usar XP como metodología de desarrollo de software, UML como lenguaje de modelado y Visual Paradigm 8.0 como herramienta CASE. Para la implementación de la herramienta se utilizará PHP 5.3 como lenguaje de programación, Symfony 2.3.1 como framework de desarrollo web y NetBeans 8.0 y PhpStorm 8.0 como entornos de desarrollo.

Capítulo 2. Análisis y Diseño

Introducción

El presente capítulo tiene como principal finalidad abordar contenidos relacionados con el análisis y diseño del sistema. Se realiza una descripción detallada de la propuesta de solución para el problema de investigación identificado. Se generan los artefactos que propone la metodología de desarrollo de *software* XP para el diseño de la aplicación. Además se definen los patrones de diseño y la arquitectura que se utilizará para el desarrollo de la solución propuesta.

2.1. Descripción de la solución propuesta

Para dar solución al problema de investigación identificado se propone el desarrollo de una herramienta *web* para la generación de datos en bases de datos PostgreSQL. El usuario una vez autenticado en el sistema podrá crear uno o varios proyectos en dependencia del rol que tenga. Podrá definir las tablas, columnas, tipos de datos, relaciones y valores a insertar en el *script sql* resultante. La herramienta será capaz de generar los tipos de datos que a continuación se mencionan, debido a que son los soportados por el motor de generación:

- *Integer*
- *Double*
- *Serial*
- *Text*
- *Character*
- *Character varying*
- *Date*
- *Timestamp*
- *Inet*
- *Macaddr*
- *Boolean*
- *Bit*
- *Uuid*

2.2. Modelo de dominio

El modelo de dominio es una herramienta de comunicación fundamental que obliga a los desarrolladores a pensar formalmente sobre el problema y permitirle validar su comprensión. Tiene como principal objetivo ayudar a entender los conceptos con los que trabajan y utilizan los usuarios y con los que deberá trabajar una aplicación. Se describe mediante diagramas de UML específicamente mediante diagramas de clases. La metodología XP no precisamente genera este artefacto, pero en la presente investigación se decide realizarlo, pues su uso agiliza y facilita el proceso de entendimiento de las clases conceptuales para aquellas personas que interactúen con la herramienta.

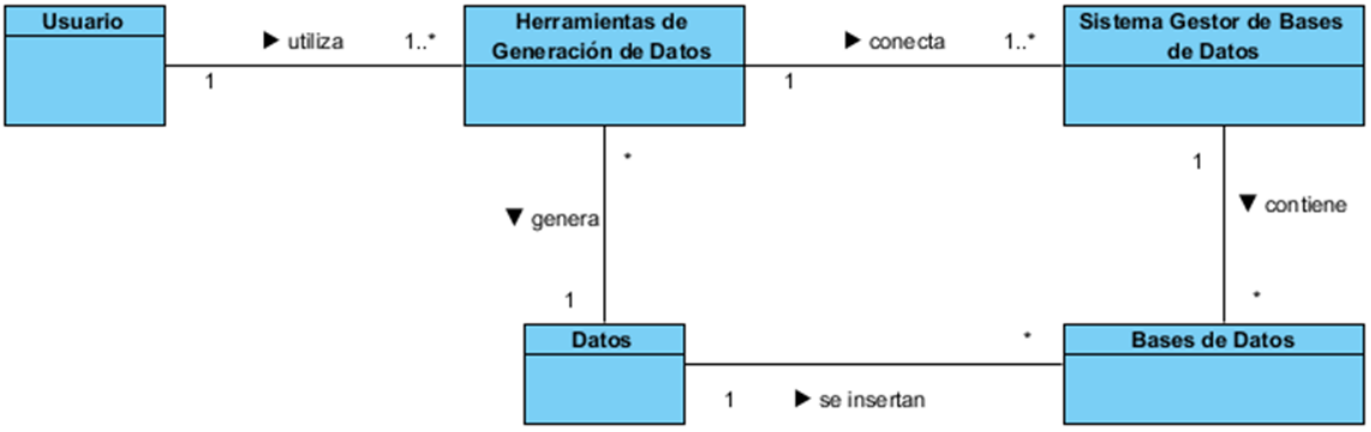


Fig.1 Modelo de dominio.

Usuario: persona que interactúa con la herramienta.

Herramientas de Generación de Datos: herramientas existentes que permiten la población de bases de datos.

Bases de Datos: bases de datos con las que trabajará la herramienta.

Sistema Gestor de Bases de Datos: sistema gestor de bases de datos sobre el cual trabajará el usuario.

Datos: volumen de datos generados por las herramientas de generación de datos que serán insertados en las bases de datos.

2.3. Historias de usuario

"La historia de usuario es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas". (Líber Batalla et al., 2006)

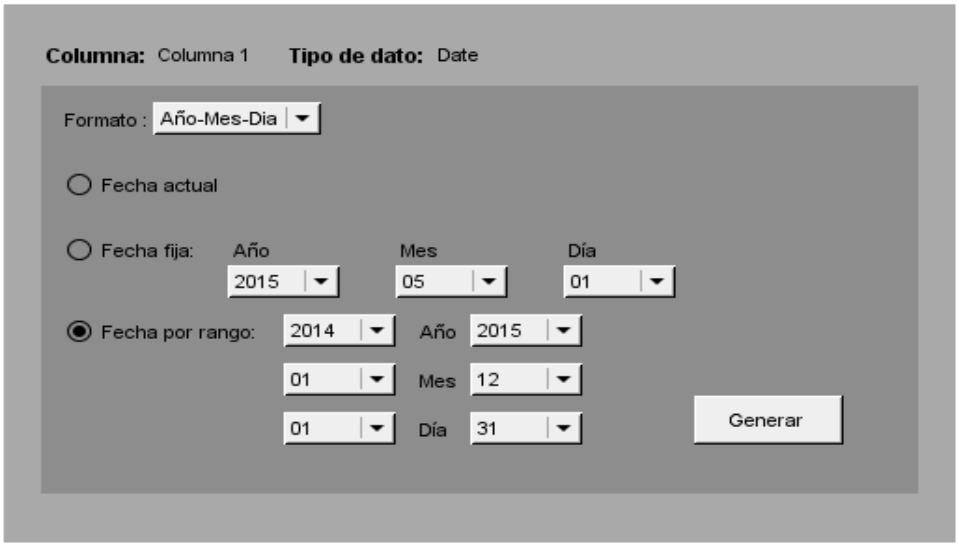
Las historias de usuario (HU) describen funcionalidades que debe incorporar un sistema y cuya implementación aporta valor al cliente. Tienen una prioridad definida por el cliente de forma tal, que indique cuáles son las más importantes para el resultado final. Se detalla la acción que realiza cada HU dejando plasmado en las observaciones los problemas que pudieran presentarse para su realización no exitosa. Se especifica de manera visual con un esbozo como quedarán conformadas mediante el prototipo de interfaces. En sentido general, su propósito principal en el avance de la herramienta es que permiten fácilmente estimar el esfuerzo necesario para implementar una nueva funcionalidad.

En la presente investigación se obtuvieron un total de 7 HU: Gestión de usuarios, Gestión de proyectos, Gestión de tablas, Gestión de columnas, Generar *script*, Generar tipos de datos y Gestión de roles. Alcanzan un grado de importancia para el desarrollo de la herramienta de muy alta, alta y media, no se encuentra ninguna que la prioridad en el negocio sea baja, puesto que es de gran importancia la realización exitosa de cada una de ellas.

Tabla 2. HU Generar tipos de datos.

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario:** Generar tipos de datos
Cantidad de modificaciones a la Historia de Usuario: 3	
Usuario: Yanet Gonzalez Dieguez	**Iteración asignada:** 2
Prioridad en negocio: alta	**Puntos estimados:** 3.0
Riesgo en desarrollo: muy alto	**Puntos reales:** 2.5

Descripción: permite darle características a los datos en dependencia de los parámetros soportados por el tipo de dato a poblar.
Observaciones:
Prototipo de interfaces:



Para ver el resto de las HU remitirse a los documentos complementarios, Artefactos, Planilla Historias de Usuario.

2.4. Lista de reserva del producto

La lista de reserva del producto (LRP) es uno de los artefactos generados por la metodología XP que contiene los Requisitos Funcionales (RF), ordenados por prioridad de implementación: muy alta, alta y media; así como los Requisitos no Funcionales (RnF) con una breve descripción de cada uno de ellos. Se refleja la estimación de cada uno de los RF, definiendo el rol que realizó la estimación. A medida que se avanza en la implementación la LRP puede ampliarse y modificarse.

Los RF son las funcionalidades del sistema. Especifican cómo debe comportarse la aplicación en situaciones particulares; sin embargo, los RnF no se refieren directamente a las funciones específicas que entrega el sistema, sino a sus propiedades. La etapa de definición de requisitos es una tarea importante a la hora de desarrollar una aplicación. Consiste en generar una definición clara de los aspectos principales del producto, cuanto más alto sea el grado de cumplimiento y aceptación por el cliente de los requisitos, más alta será la calidad del sistema desarrollado.

Tabla 3. Lista de reserva del producto.

Ítem *	Descripción	Estimación	Estimado por
Prioridad: muy alta			
1	Registrar usuario	0.5	Analista
2	Autenticar usuario	0.5	Analista
3	Editar usuario	1	Analista
4	Eliminar usuario	1	Analista
5	Insertar proyecto	1	Analista
6	Eliminar proyecto	1	Analista
7	Editar proyecto	1	Analista
8	Crear tabla	0.5	Analista
9	Eliminar tabla	0.5	Analista
10	Editar tabla	0.5	Analista
11	Crear relación	0.5	Analista
12	Eliminar relación	0.5	Analista
13	Editar relación	0.5	Analista
14	Insertar columna	1	Analista
15	Eliminar columna	1	Analista
16	Editar columna	1	Analista
Prioridad: alta			
17	Importar script	1	Analista
18	Generar tipo de dato integer	0.3	Analista
19	Generar tipo de dato double	0.2	Analista

20	Generar tipo de dato serial	0.3	Analista
21	Generar tipo de dato text	0.1	Analista
22	Generar tipo de dato character	0.1	Analista
23	Generar tipo de dato character varying	0.3	Analista
24	Generar tipo de dato date	0.3	Analista
25	Generar tipo de dato timestamp	0.3	Analista
26	Generar tipo de dato inet	0.1	Analista
27	Generar tipo de dato macaddr	0.3	Analista
28	Generar tipo de dato boolean	0.1	Analista
29	Generar tipo de dato bit	0.3	Analista
30	Generar tipo de dato uuid	0.3	Analista
31	Exportar script	2	Analista
Prioridad: media			
32	Enviar petición	1	Analista
33	Eliminar petición	1	Analista
34	Cambiar rol	1	Analista
Requisitos no funcionales			
1	**Apariencia:** la aplicación contará con una interfaz sencilla, con colores suaves, entre ellos verdes y azules.		
2	**Legales: p**ara el desarrollo de la herramienta se estará usando herramientas de software libre con licencia GNU/GPL.		
3	**Persistencia:** la información del sistema debe almacenarse en bases de datos con el objetivo de poder hacer análisis de la misma en cualquier momento, sólo borrable por el usuario.		
4	**Usabilidad:** la aplicación podrá ser utilizada por personal que tenga conocimientos elementales de informática y bases de datos.		
5	**Seguridad:** para poder acceder al sistema el usuario necesita autenticarse y su acceso será restringido por un tipo de rol.		
6	**Portabilidad:** será un sistema multiplataforma, se podrá disponer del mismo tanto en el sistema operativo Linux como Windows.		
7	**Hardware:** Las PC que sean clientes de la aplicación deben tener las siguientes		

	prestaciones mínimas: Microprocesador: 500 MHz Memoria RAM: 128 MB Tarjeta de red. Los servidores de la aplicación deben tener las siguientes prestaciones mínimas: Microprocesador: 3.00 GHz Memoria RAM: 1 GB Tarjeta de red.	
8	**Software:** sistema operativo Windows (XP o superior) y Linux (cualquier distribución). Navegador web Mozilla Firefox o Internet Explorer. Se requiere que el servidor tenga instalado PostgreSQL 9.3.	

2.5. Plan de iteraciones

El plan de iteraciones define exactamente cuáles HU serán implementadas en cada iteración. Se puede establecer una arquitectura del sistema en la primera iteración y utilizarla durante la trayectoria del proyecto. Se logra escogiendo las HU correctas para la creación de la arquitectura; sin embargo, esto no siempre es posible, ya que es el cliente quien decide qué historias se implementarán en cada iteración. El sistema estará listo para su utilización al final de la última iteración. Durante su elaboración debe tomarse en cuenta las HU no abordadas, la velocidad del proyecto y las tareas no terminadas en la iteración anterior. Es necesario crearlo una vez definidas las HU con el objetivo de definir las tareas con mayor prioridad y establecer los tiempos de implementación.

Tabla 4. Plan de iteraciones.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	En la primera iteración se implementarán las HU 1, 2, 3 y 4, las cuales son de prioridad muy alta, por tanto tienen gran incidencia sobre el sistema.	1, 2, 3, 4	12 semanas

Iteración 2	En la segunda iteración serán implementadas las HU 5 y 6 que son de prioridad alta, es decir, desempeña un buen papel en el correcto funcionamiento del sistema.	5, 6	6 semanas
Iteración 3	En la tercera iteración se implementará la HU 7 que es de prioridad media, es decir, no tiene mucha incidencia sobre el funcionamiento del sistema, pero se tiene en cuenta para la asignación de roles a los usuarios.	7	3 semanas

2.6. Arquitectura de software

"En su forma más simple, la arquitectura es la estructura u organización de los componentes del programa (módulos), la manera en que éstos componentes interactúan y la estructura de datos que utilizan los componentes". (Pressman, 2011)

La arquitectura de *software* se puede definir a grandes rasgos, como una vista del sistema que incluye sus componentes principales, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que estos interactúan y se coordinan para alcanzar la misión del sistema. Define estilo o una combinación de estilos para una solución informática y es esencial para el éxito o fracaso de un proyecto.

2.6.1. Estilos arquitectónicos

"El estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema". (Pressman, 2011)

Los estilos arquitectónicos son artefactos de ingeniería importantes, porque definen clases de diseño junto con las propiedades conocidas asociadas a ellos. Catalogan las formas básicas posibles de estructuras de software y definen los patrones posibles de las aplicaciones. Los estilos de Llamada y Retorno permiten construir una estructura de programa relativamente fácil de modificar y ajustar a gran escala. El patrón Modelo-Vista-Controlador (MVC) como uno de los estilos de Llamada y Retorno fue el empleado en el desarrollo de la aplicación.

2.6.2. Patrón arquitectónico

"Un patrón arquitectónico, al igual que un estilo, impone una transformación en el diseño de una arquitectura. Sin embargo, un patrón difiere de un estilo en varios elementos fundamentales: 1) el alcance de un patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura, 2) un patrón impone una regla sobre la arquitectura, pues describe la manera en que el software manejará algún aspecto de su funcionalidad al nivel de la infraestructura, 3) los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura". (Pressman, 2011)

Los patrones arquitectónicos son patrones de *software* que ofrecen soluciones a problemas de arquitectura de *software* en la ingeniería de *software*. Se utilizan para expresar una estructura de organización base, proporcionando un conjunto de subsistemas predefinidos, especificando directrices que determinan la interacción y relaciones entre ellos. Los beneficios de su utilización van desde la imposición de decisiones tempranas en el desarrollo hasta la reutilización. Como se había mencionado anteriormente el patrón arquitectónico utilizado fue el MVC para lograr una arquitectura más robusta y consistente de la aplicación. Con su uso es mucho más sencillo agregar múltiples representaciones de datos o información. Brinda la posibilidad de dar mantenimiento en caso de errores y ofrece formas sencillas para probar el funcionamiento correcto de la aplicación. Predomina en el *framework* de desarrollo Symfony utilizado para la confección del presente trabajo. Consiste en separar la lógica de control, la interfaz de usuario y los datos de una aplicación en tres componentes distintos: el modelo, la vista y el controlador.

Modelo

El paquete modelo encapsula el núcleo funcional y los datos involucrados. Responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Contiene todas las clases que tiene el código relacionado con el acceso a los datos para que sea lo más genérico posible y se pueda reutilizar.

Vista

El paquete vista maneja la visualización de la información, o sea, muestra la información al usuario. Pueden existir múltiples vistas del modelo cada una teniendo asociado un componente controlador. Contiene todas las clases que tiene el código, representando la parte que será visualizada por el usuario.

Controlador

El paquete controlador interpreta las acciones del usuario de teclado y ratón, informando al modelo y a la vista para cambiar apropiadamente sus estados. Contiene las clases que ejecutan la lógica de la aplicación, realiza llamadas al modelo para obtener los datos y se los envía a la vista para que los muestre al usuario. Recibe las entradas, normalmente como eventos y los eventos son traducidos a solicitudes de servicio, bien para el modelo bien para la vista.

Tanto la vista como el controlador dependen del modelo; sin embargo, el modelo no depende de la vista. La separación permite que el modelo sea construido y probado independientemente de la presentación visual. La separación entre vista y controlador es secundaria en muchas aplicaciones; sin embargo en las aplicaciones *web* la vista (el navegador) y el controlador (los componentes del lado servidor) están bien definidos.

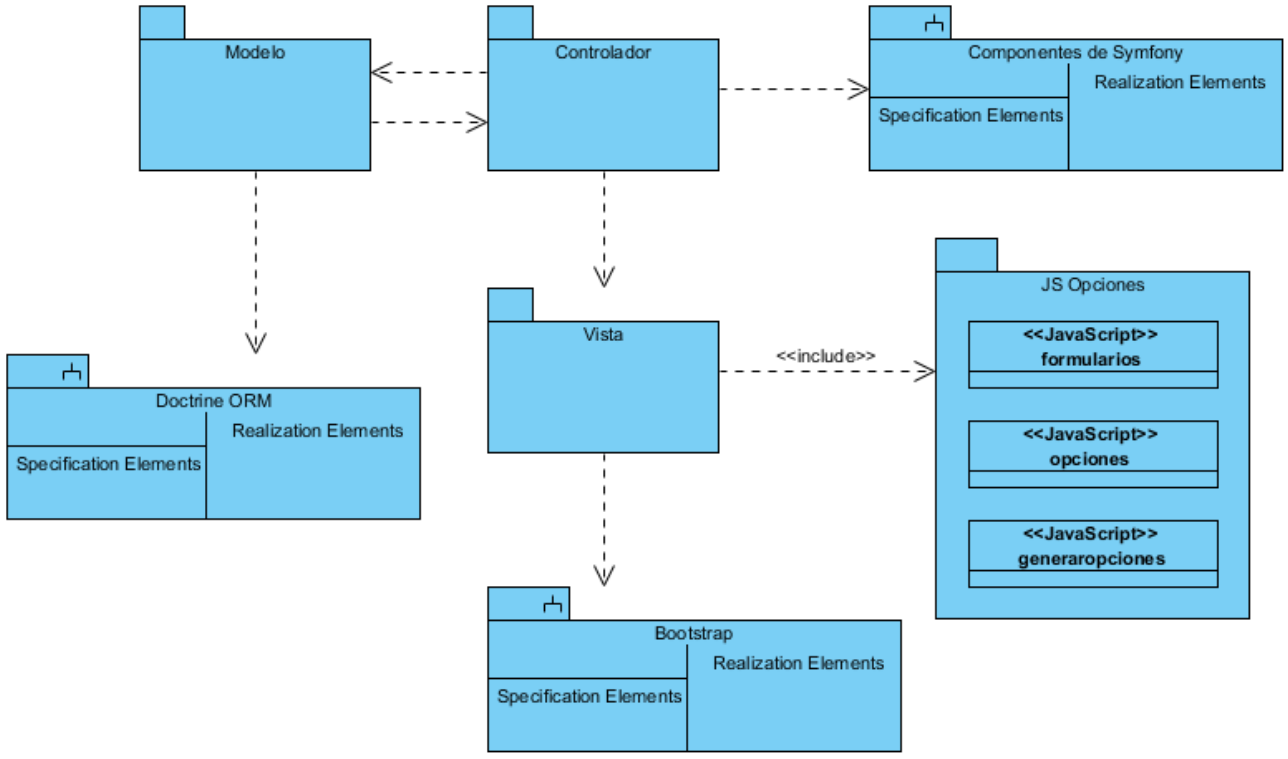


Fig.2 Vista lógica del sistema.

2.7. Tarjetas CRC

Las tarjetas CRC (Clase-Responsabilidades-Colaboraciones) trabajan con una metodología basada en objetos. Su objetivo es hacer mediante tarjetas un inventario de las clases que se necesitarán para implementar el sistema y la forma en que van a interactuar. Las tarjetas CRC presentan características tales como, la identificación de clases y asociaciones que participan en el diseño del sistema, la obtención de las responsabilidades que debe cumplir cada clase y el establecimiento de como una clase colabora con otras clases para cumplir con sus responsabilidades. Para el presente trabajo se confeccionaron un total de 5 tarjetas CRC: UsuarioController, AdminController, ProyectoController, PostgreController y GenerarController.

Tabla 5. Tarjeta CRC para la clase GenerarController.

Tarjeta CRC	
Clase: GenerarController	
Responsabilidades	**Colaboraciones**
• Crea un nuevo script sql. • Insertar esquemas al script. • Insertar tablas al script. • Insertar columnas al script. • Insertar datos al script. • Insertar relaciones al script.	• ProyectoController

Para ver el resto de las tarjetas CRC remitirse a los documentos complementarios, Artefactos, Planilla Tarjetas CRC.

2.8. Diagrama de clases del diseño

Un diagrama de clases del diseño es un diagrama que se utiliza para modelar la vista estructural de un sistema. Describe gráficamente las especificaciones de las clases, permitiendo visualizar las relaciones entre ellas. Se puede utilizar siempre y cuando contribuya para el mejoramiento de su comprensión,

porque es una guía importante para los desarrolladores a la hora de implementar. Es una herramienta esencial durante el proceso de análisis y diseño del sistema. Se decide incorporar a la investigación en aras de lograr una descripción más detallada y fácil de entender de la aplicación.

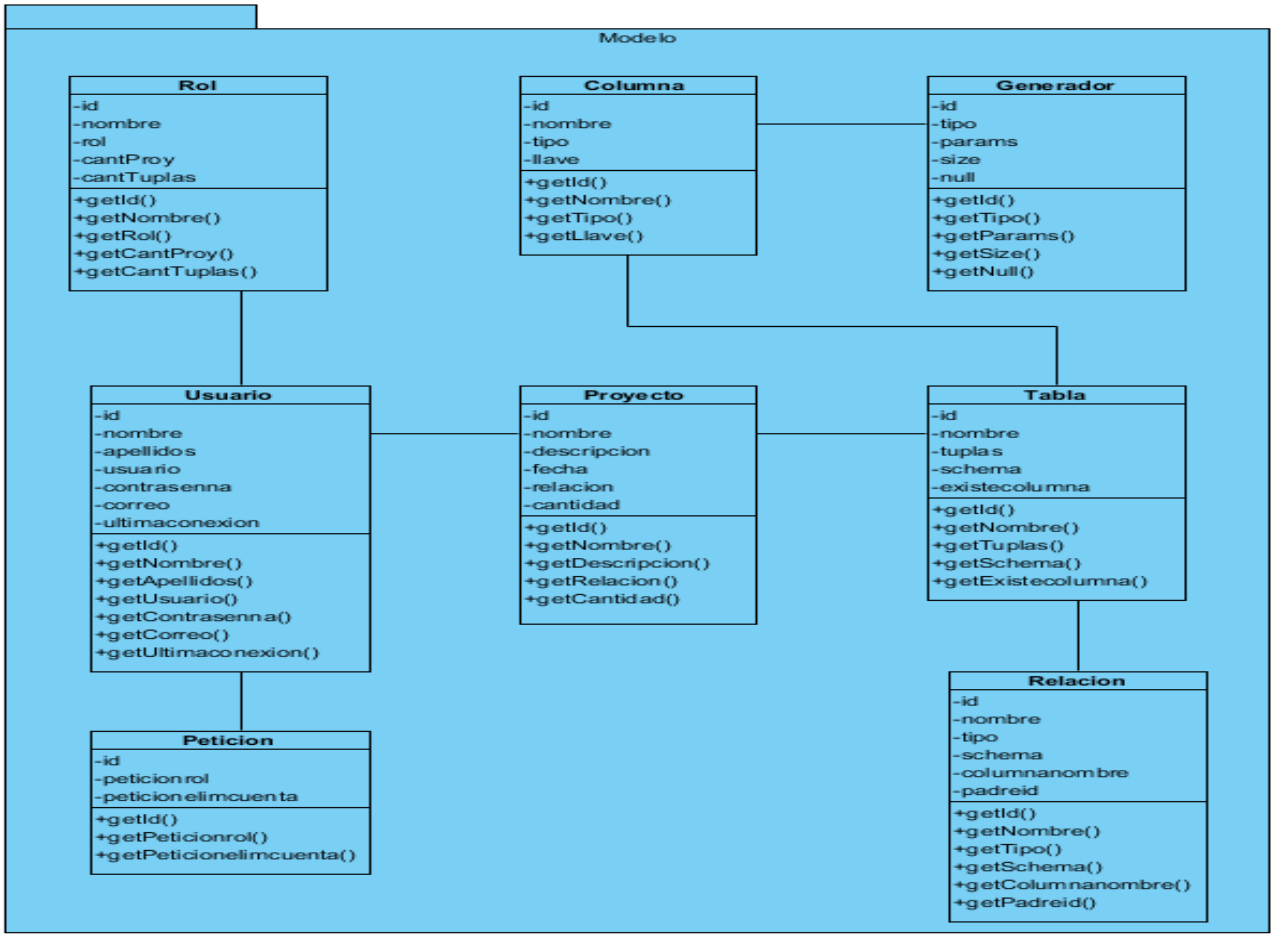


Fig.3 Diagrama de clases del diseño (Modelo).

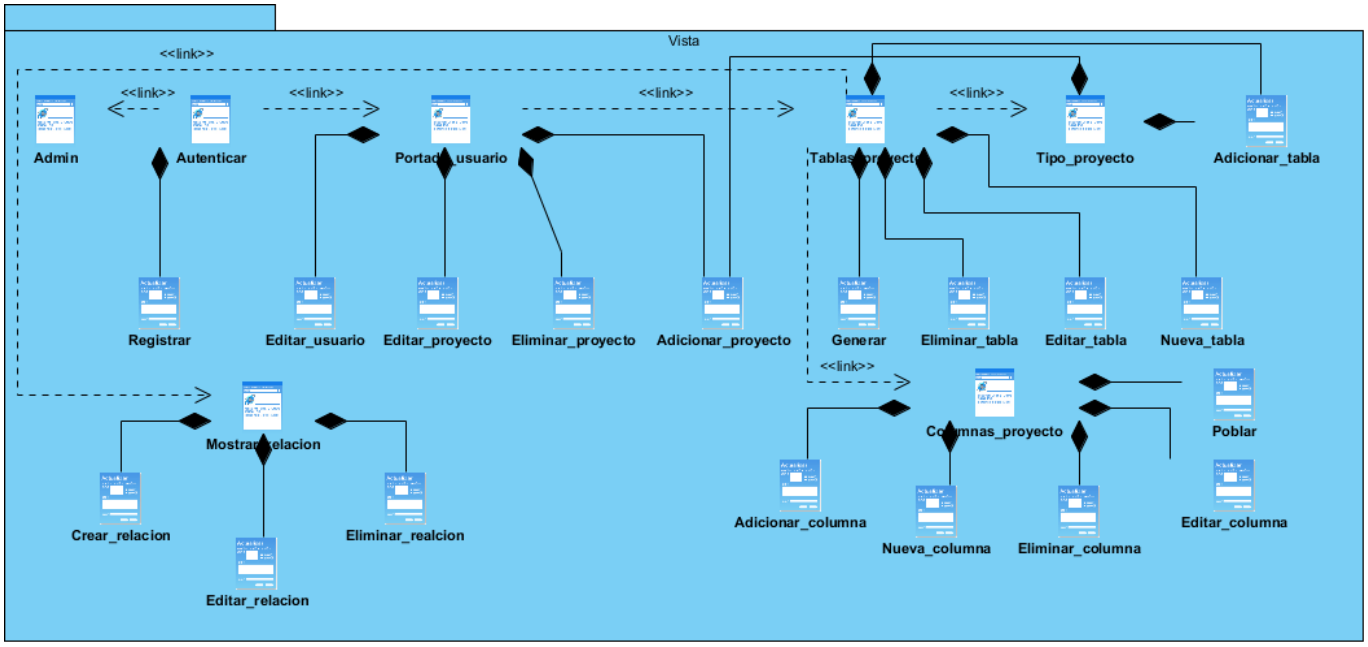


Fig.4 Diagrama de clases del diseño (Vista).

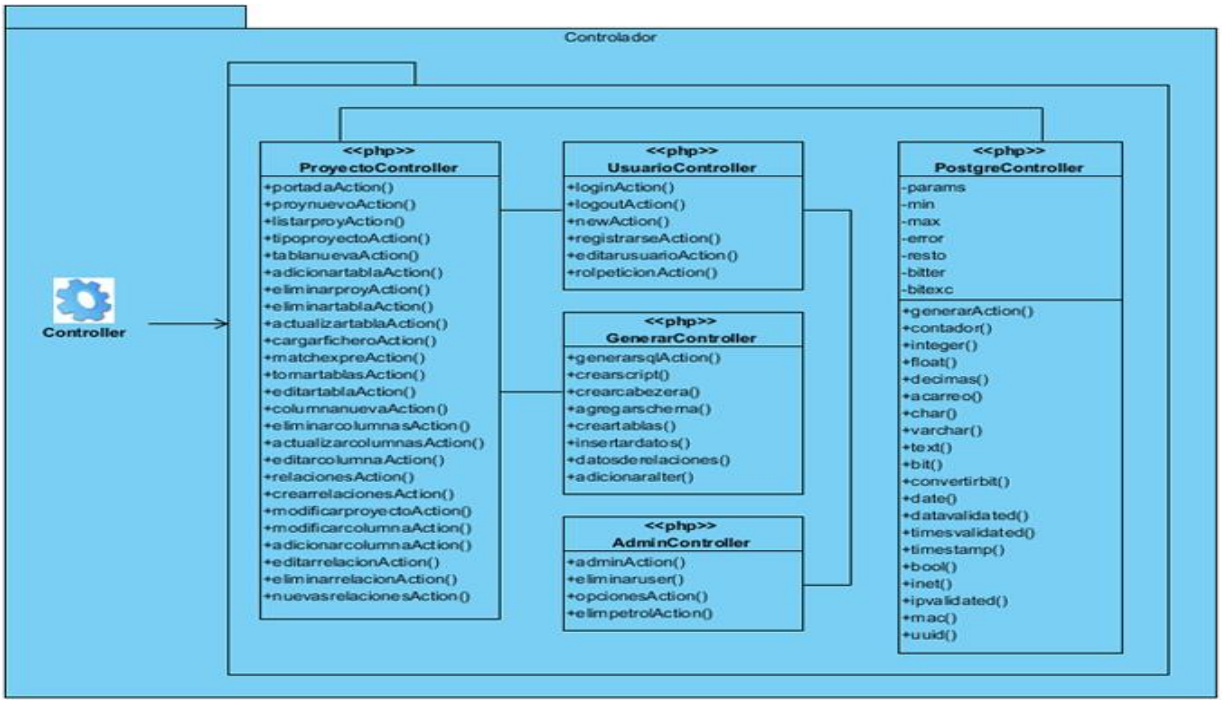


Fig.5 Diagrama de clases del diseño (Controlador).

2.9. Patrones de diseño

"Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos. Un patrón de diseño puede considerarse como un documento que define una estructura de clases que aborda una situación particular". (Kioskea.net, 2015)

Los patrones de diseño contribuyen a reutilizar diseño gráfico identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una cierta cantidad de situaciones. Brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares. Con su uso se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

2.9.1. Patrones GRASP

"Los patrones GRASP (Patrones de Software para la Asignación General de Responsabilidad) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones". (Grosso, 2011)

Los patrones GRASP constituyen un apoyo para la enseñanza, pues ayudan a entender el diseño de objetos. Se ha tenido en cuenta para la realización de la herramienta los siguientes patrones GRASP:

Creador

El patrón creador guía la asignación de responsabilidades y ayuda a identificar quién debe ser el responsable de la creación de objetos. Su propósito fundamental es encontrar un creador que se deba conectar con el objeto producido en cualquier evento. El uso del patrón se evidencia en la clase ProyectoController, ya que tiene las cualidades y responsabilidades para crear instancias de objetos.

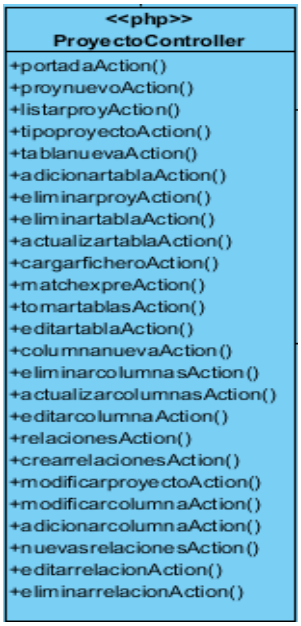


Fig.6 Fragmento del diagrama de clases del diseño donde se evidencia el patrón creador.

Experto

El patrón experto asigna una responsabilidad al experto en información, la clase que cuenta con la información necesaria para realizar la tarea. Es un principio básico que suele utilizarse en el diseño orientado a objetos. Se evidencia en la clase PostgreController, debido a que es la única que contiene toda la información acerca de los tipos de datos que se generan y de esta forma se convierte en la clase experta.

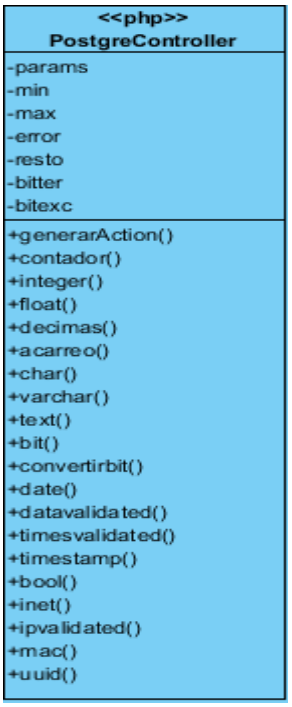


Fig.7 Fragmento del diagrama de clases del diseño donde se evidencia el patrón experto.

Controlador

El patrón controlador asigna la responsabilidad del manejo de los eventos del sistema a clases específicas. Se puede evidenciar su uso en la clase ProyectoController, la cual es la encargada de controlar y manejar todas las tareas.

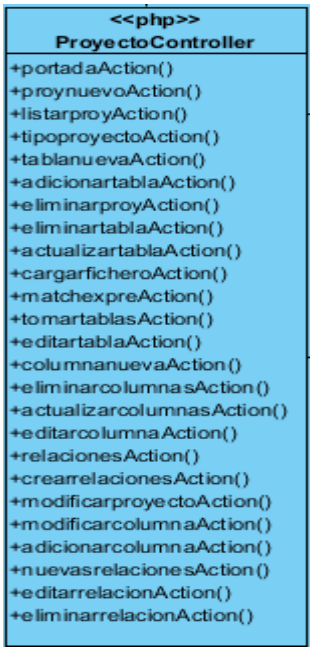


Fig.8 Fragmento del diagrama de clases del diseño donde se evidencia el patrón controlador.

2.9.2. Patrones GOF

"Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular (GOF)". (Prieto, 2009)

Los patrones GOF (por sus siglas en inglés *Gang of Four*) recopilan una serie de patrones de diseños agrupados en tres categorías: creación, estructura y comportamiento. Se ha tenido en cuenta para la realización de la herramienta el siguiente patrón GOF:

Estructurales: describen las clases y objetos que pueden ser combinados para establecer grandes estructuras y proporcionar nuevas funcionalidades.

- **Facade (Fachada):** proporciona una interfaz unificada y de alto nivel para un conjunto de interfaces de un subsistema lográndose que sea más fácil de usar.

El uso del patrón se evidencia en la clase ProyectoController, debido a que unifica las clases visuales y las muestra en una sola interfaz.

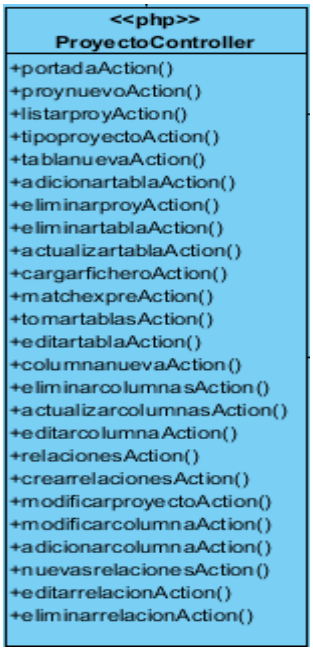


Fig.9 Fragmento del diagrama de clases del diseño donde se evidencia el patrón fachada.

2.10. Diagrama de despliegue

Un diagrama de despliegue define cómo se distribuye físicamente un sistema. Muestra su arquitectura desde el punto de vista del despliegue (distribución) de los artefactos del *software* en los destinos de despliegue. Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Los destinos de despliegue están generalmente representados por un nodo que es o bien de los dispositivos de *hardware* o bien algún entorno de ejecución de *software*. Los nodos pueden ser conectados mediante conexiones de comunicación, generalmente mediante protocolos de comunicación TCP/IP, HTTP o HTTPS.

Fig.10 Diagrama de despliegue.

Conclusiones del capítulo

Se identificaron un total de 7 HU, de las cuales se definieron 34 RF, planificando su implementación para 3 iteraciones comenzando por las HU de prioridad muy alta en el negocio y 8 RnF. Se describe la arquitectura a través del uso del patrón arquitectónico MVC. Se confeccionaron 5 tarjetas CRC teniendo en cuenta las clases necesarias para implementar el sistema. Fueron identificados varios patrones de diseño, de los cuales como patrones GRASP: Creador, Experto y Controlador; y como patrón GOF: Fachada. La realización del diagrama de despliegue propició una visión de cómo está distribuido el sistema físicamente, quedando representado por 3 nodos de procesamiento: 1 pc cliente, 1 servidor de aplicaciones *web* y 1 servidor de bases de datos.

Capítulo 3. Implementación y Prueba

Introducción

El contenido que se aborda en el capítulo está relacionado con la descripción de la implementación del sistema para darle solución a las historias de usuario definidas en el capítulo anterior. Se realizan las tareas de ingeniería, se definen los estándares de codificación y se especifican las pruebas a las que fue sometida la aplicación para validar que cumple con las expectativas del usuario final.

3.1. Implementación

La etapa de implementación es esencialmente el proceso de poner en práctica el diseño, o sea, es donde efectivamente se programa el sistema. Un diseño de alta calidad por lo general conducirá a una implementación efectiva. Tiene como principal objetivo desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código.

Para desarrollar la implementación el motor de generación es la parte esencial para el correcto funcionamiento del sistema. Se hace necesario la creación del propio motor de la herramienta sobre la guía del *datafiller* evitando su utilización, debido a que fue realizado en un lenguaje desconocido por los programadores y sólo genera datos de forma aleatoria. Por lo que el equipo de desarrollo decidió a partir de su propio lenguaje de programación realizar el sistema apoyado en las funcionalidades brindadas por el *script*, permitiendo agregar otras formas de generación no sólo la soportada por el *datafiller*, sino también mediante listas o archivos externos.

3.1.1. Tareas de la programación

Para llevar a cabo la implementación se desglosa en tareas de la programación (TP) las HU definidas en el capítulo anterior; siendo así más sencillo el desarrollo de la aplicación logrando una programación más eficiente. Las TP son un conjunto de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con la especificación de una solución. Se utilizan para describir las tareas que se realizan sobre el proyecto que pueden ser de desarrollo, corrección y mejora. Para que las HU definidas se puedan lograr desarrollar con la calidad requerida, se desglosan en TP y se asignan a los programadores para que en cada iteración sean implementadas. De esa forma los programadores a la

hora de implementar las HU obtienen una mejor visión.

Tabla 6. TP Insertar proyecto.

Tarea de la Programación	
Número de la tarea: 5	**Número de la historia de usuario:** 2
Nombre de la tarea: Insertar proyecto	
Tipo de tarea: desarrollo	**Puntos estimados:** 1.0
Fecha inicio: 09/02/2015	**Fecha fin:** 16/02/2015
Programador responsable: Eddy Roy Velázquez Parra	
Descripción: se crea una entidad de tipo proyecto identificándose el nombre, descripción, cantidad de proyectos creados, si hay o no relaciones entre las tablas y la fecha en que se inserta el proyecto.	

Para ver el resto de las TP remitirse a los documentos complementarios, Artefactos, Planilla Tareas de la Programación.

3.1.2. Estándares de codificación

El código debe ser desarrollado siguiendo estándares de desarrollo para facilitar su lectura y la modificación por cualquier miembro del equipo de desarrollo. El propósito fundamental de los estándares de codificación es que la herramienta tenga una arquitectura y un estilo consistente, independiente del autor con lo cual el sistema resulte fácil de entender en el menor tiempo posible y fácil de mantener. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final en la aplicación, brindando la posibilidad de asegurarse que los programadores trabajen de forma coordinada. Si se aplica de forma continua un estándar de codificación bien definido y luego se efectúan revisiones del código, caben muchas posibilidades de que un proyecto de software se convierta en un sistema sencillo, garantizando un mantenimiento óptimo del código por parte de los programadores. A continuación se presenta un fragmento de los estándares definidos en la implementación.

- **Identación**

La unidad de identado es de 4 espacios. El uso de la tabulación debe ser evitado, pues no existe un estándar que determine con precisión el ancho que va a producir la tabulación.

```
public function tablasAction($id) {
    $em = $this->getDoctrine()->getManager();
    $p = $em->getRepository('ProyectoBundle:Tablas')->findOneBy(array('id' => $id));
    $proy = $p->getProyecto_id();
    $tab = $em->getRepository('ProyectoBundle:Tablas')->findBy(array('proyecto_id' => $proy));
    $entity = $em->getRepository('ProyectoBundle:Proyecto')->findOneBy(array('id' => $proy));
    return $this->render('ProyectoBundle:Paginas:tablasproyecto.html.twig', array('result' => $tab, 'proyecto' => $entity));
}
```

Fig.11 Ejemplo de identación evidenciado en la implementación.

- **Comentarios**

Es conveniente dejar información que pueda ser leída tiempo después por personas que necesiten entender lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea. Se debe reservar los comentarios de bloques para la documentación formal o para comentar fragmentos de código.

```
/**
 * Permite regresar a la portada principal
 * Llamada a listarproy
 * @return portada
 */
public function portadaAction() {
    $this->listarproyAction();
    return $this->render('ProyectoBundle:Principal:portada_user.html.twig');
}
```

Fig.12 Ejemplo de comentarios evidenciado en la implementación.

- **Declaración de variables**

Cada variable debe de ser declarada en una línea. El nombre de las variables debe comenzar con letra minúscula y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula.

```
var $params;
var $min;
var $max;
var $error;
```

Fig.13 Ejemplo de declaración de variables evidenciado en la implementación.

- **Declaración de funciones**

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre el paréntesis izquierdo y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que las variables.

```
public function tipoproyectoAction($id) {
    $em = $this->getDoctrine()->getManager();
    $proyecto = $em->getRepository('ProyectoBundle:Tablas')->findBy(array('proyecto_id' => $id));
    if ($proyecto == NULL) {
        return $this->render('ProyectoBundle:Paginas:tipoproyecto.html.twig', array('proyecto_id' => $id));
    }
    $entity = $em->getRepository('ProyectoBundle:Proyecto')->find(array('id' => $id));
    return $this->render('ProyectoBundle:Paginas:tablasproyecto.html.twig', array('result' => $proyecto, 'proyecto' => $entity));
}
```

Fig.14 Ejemplo de declaración de funciones evidenciado en la implementación.

- **Sentencia return**

Una sentencia return no debe utilizar paréntesis “()” alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminada con un punto y coma.

```
public function cajatablaAction($id) {
    return $this->render('ProyectoBundle:Paginas:cajatabla.html.twig', array('proyecto_id' => $id));
}
```

Fig.15 Ejemplo de la sentencia return evidenciado en la implementación.

- **Sentencia if**

La sentencia if siempre lleva “{}”.

```
if (move_uploaded_file($_FILES['cargarScript']['tmp_name'], $uploadfile)) {
    $file = file_get_contents($uploadfile);
    $this->matchExpreAction($file, $id);
    return $this->redirect($this->generateUrl('portada'));
}
```

Fig.16 Ejemplo de la sentencia if evidenciado en la implementación.

- **Sentencia for**

La sentencia for posee la siguiente estructura para cualquier lenguaje, sólo varía la manera en escribirla: for ([<inicio>] ; [<condición>] ; [<incremento>]) <sentencia>

```
for ($i; $i < count($gener); $i++) {
    $em->remove($gener[$i]);
    $em->flush();
}
```

Fig.17 Ejemplo de la sentencia for evidenciado en la implementación.

- **Sentencia while**

La sentencia while siempre lleva “{}”.

```
while ($petN = $request->get('nombre' . $cont) != NULL && $petT = $request->get('tuplas' . $cont) != NULL) {
    $petN = $request->get('nombre' . $cont);
    $petT = $request->get('tuplas' . $cont);
    if ($petT <= 0 || $petT > 1000) {
        $petT = 1000;
    }
    $entity = new Tablas($petN, $petT, $id);
    $em = $this->getDoctrine()->getManager();
    $em->persist($entity);
    $em->flush();
    $cont = $cont + 1;
}
```

Fig.18 Ejemplo de la sentencia while evidenciada en la implementación.

- **Espacios en blanco**

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionados. Los espacios en blanco pueden ser (o no deben ser) utilizados en las siguientes circunstancias:

- No se debe utilizar un espacio en blanco entre el identificador de una función y el paréntesis que abre la lista de parámetros. Esto ayuda a distinguir entre palabras reservadas y llamadas a funciones.
- Para cualquier operador binario excepto el punto “.”, el paréntesis que abre “(” y el corchete que abre “[” todos deben ser separados por un espacio entre operandos y operador.
- Debe dejarse un espacio luego de cada coma “,”.

```
public function tablanuevaAction($id) {
```

Fig.19 Ejemplo de los espacios en blanco evidenciado en la implementación.

- **Nombre de las clases**

Las clases comenzarán con mayúscula, si la clase es compuesta empezarán con mayúsculas las dos letras iniciales de cada palabra y estarán unidas.

```
class Proyecto {
```

Fig.20 Ejemplo del nombre de las clases evidenciado en la implementación.

3.2. Pruebas de software

La comprobación del funcionamiento de un producto de *software* es el proceso de demostrar si satisface los requisitos iniciales, es decir, si lo que se ha especificado es lo que realmente el usuario quería. Permite aumentar la calidad de la aplicación y la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones. Una de las ventajas principales que ofrece la metodología XP, es proporcionar al cliente la posibilidad de concretar las funcionalidades de las HU y verificarlas en el proceso de pruebas, logrando así el correcto funcionamiento del sistema.

Las pruebas de *software* son un conjunto de actividades que se realizan con el objetivo de encontrar errores en la implementación o calidad de un sistema. Juegan un papel fundamental cuando se desea obtener un producto de alta calidad, donde se verifica que realmente cumple con las expectativas del usuario final. Constituyen un factor fundamental para garantizar la calidad del *software*, representan una revisión final del diseño y las especificaciones.

3.2.1. Niveles de prueba

La prueba es aplicada con diferentes objetivos y en disímiles escenarios o niveles de prueba, los que deben fluir de forma organizada y se considera que deben realizarse todos para garantizar la calidad. Los niveles de prueba son diferentes ángulos de verificar y validar en determinados momentos el ciclo de vida de un *software*. En el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL se aplicarán las pruebas que abarcan los siguientes niveles:

- Desarrollador: detecta errores en la implementación de requisitos.

- Aceptación: detecta fallas en la implementación del sistema.

Después de un análisis se determina que los niveles de prueba a emplear durante el proceso de pruebas son desarrollador y aceptación, debido a que están destinadas a verificar que el *software* construido en las iteraciones realizadas cumple con las funcionalidades solicitadas por el usuario y está listo para ser utilizado en el ambiente del cliente. Aseguran que las funcionalidades del sistema cumplen con lo que se espera de ellas, brindándole al cliente satisfacción del producto esperado. Son definidas y diseñadas por el cliente, permitiendo comprobar el funcionamiento del sistema, posibilitando que los programadores estén al tanto de lo que falta por realizar para que el cliente una vez terminado certifique que el sistema es válido para él.

3.2.2. Tipos de prueba

Teniendo en cuenta los niveles planteados anteriormente se seleccionan los siguientes tipos de prueba para asegurar el correcto funcionamiento del sistema:

- Pruebas de función: las pruebas de función tienen como objetivo asegurar el trabajo apropiado de los RF; incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

- Pruebas de aceptación: las pruebas de aceptación determinan por parte del cliente la aceptación o rechazo del sistema desarrollado, definiendo dos tipos: alfa y beta para posibilitar el descubrimiento de errores que sólo el usuario final podría detectar. El usuario del *software* no está interesado en saber cómo se desarrolló o cómo funciona el código fuente; por esto cuando se construye un *software* para un cliente, se lleva a cabo una serie de pruebas de aceptación para permitir que el cliente valide todos los requisitos. Lo que importa para el usuario es la funcionalidad del *software*; por lo tanto en las pruebas alfa y beta el código fuente no está probado y la atención se centra en la funcionalidad del *software* de acuerdo con las necesidades del usuario.

"La prueba alfa se lleva a cabo, por un cliente, en el mismo lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado. La prueba beta es la que se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa el desarrollador no está presente, así la prueba beta es una aplicación en vivo del software en un entorno

que no puede ser controlado por el desarrollador. El cliente registra todos los problemas que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador". (Lastra, 2010)

Se decide emplear en las pruebas de aceptación el tipo alfa, debido a que permite que los usuarios puedan conocer la solución propuesta, realizar pruebas para identificar defectos o dar retroalimentación sobre la implementación de sus funciones. Es posible que a raíz de estas pruebas la aplicación sufra cambios importantes en su estructura tanto interna como de interface de usuario.

3.2.3. Métodos de prueba

Cualquier proyecto que se trace una estrategia de prueba debe contar con métodos de prueba para la aplicación de cada una de las pruebas. Tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Proporcionan un mecanismo de ayuda para asegurar la calidad con la que cuenta un *software* y la mayor probabilidad de descubrimiento de errores que tiene el mismo. Los métodos tradicionales son:

Prueba de caja blanca

"La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba". (Pressman, 2011)

No se decidió utilizar el método de caja blanca, debido a que este método calcula la complejidad ciclomática para de esta forma saber el grado de complejidad y generalmente se utiliza cuando se programan algoritmos. No tiene sentido hacérselo a la aplicación, porque como el sistema no usa ningún algoritmo al calcularse la complejidad de algún método sería una complejidad baja, o sea, los métodos no son complejos.

Prueba de caja negra

"La prueba de caja negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa". (Pressman, 2011)

Para desarrollar pruebas de caja negra existen varias técnicas, entre ellas se encuentran:

- Partición equivalente: divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. Se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse.
- Análisis de valores límites (AVL): es mayor el número de errores que se presenta en los límites del dominio de entrada que en el centro; por ello se desarrolló el AVL como técnica de prueba. El AVL lleva a una selección de casos que prueba los valores límites.
- Prueba de tabla ortogonal: la prueba de tabla ortogonal se aplica en problemas en los cuales el dominio de entrada es relativamente pequeño, pero demasiado grande para una prueba exhaustiva. (Pressman, 2011)

Se decide utilizar el método de caja negra, debido a que permite obtener un conjunto de condiciones de entradas que permiten ejercitar totalmente todos los RF del programa. Se lleva a cabo sobre la interfaz del *software* y tiene como objetivo demostrar que las funciones del *software* son operativas, que las entradas se acepten de forma adecuada y se produzca un resultado correcto, teniendo en cuenta que la integridad de la información externa se mantenga. El método de caja negra ahorra tiempo, es decir, prueba el *software* de forma efectiva determinando si funciona correctamente. Específicamente se utilizará la técnica partición equivalente del método de caja negra, debido a que permite identificar claramente las entradas y salidas; y estudiar las relaciones que existen entre ellas. Además, cada una de las clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada.

3.3. Diseño de casos de prueba basados en historias de usuario

Los diseños de casos de prueba se utilizan para verificar que los componentes del sistema interaccionan entre sí de la forma más apropiada posible, ya que se diseñan un conjunto de casos de prueba que permiten alcanzar los objetivos trazados. Se trata de diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo. El propósito de un caso de prueba es especificar una forma de probar el sistema, incluyendo las entradas con las que se prueba, los resultados esperados y las condiciones bajo las que ha de probarse. En la prueba de caja negra, los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la

entrada se acepta de forma adecuada y que se produce una salida correcta. Para validar las funcionalidades del sistema se diseñaron un total de 7 casos de prueba correspondientes a las 7 HU identificadas.

Tabla 7. Descripción de las variables del caso de prueba: Gestión de proyectos.

No	Nombre de campo	Clasificación	Valor nulo	Descripción
Variable 1	Nombre	Campo de texto	No	Este campo de texto admite cualquier combinación de letras mayúsculas y minúsculas.
Variable 2	Descripción	Campo de texto	No	Este campo de texto admite cualquier combinación de letras mayúsculas y minúsculas.

Tabla 8. Caso de prueba: Gestión de proyectos (Editar proyecto).

Escenario	Descripción del escenario	Nombre	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Editar un proyecto correctamente.	Permite editar los datos de los proyectos creados en el sistema.	V Nuevo V Otro NA	V Es un proyecto NA V Para las tablas	Una vez seleccionado el proyecto deseado para editar algunos de sus datos el sistema actualiza dichos datos.	1-Seleccionar opción Adicionar.2-Llenar los campos requeridos.3-Aceptar.4-Se selecciona el proyecto a editar.5-Cambia los campos que desea editar.6-Aceptar.7-Se actualizan los cambios realizados.

EC 1.2 Editar un proyecto con los campos incorrectos.	Se verifica si todos los datos están escritos correctamente.	I Otro	NA Para la historia	Al llenar un campo incorrectamente alerta al usuario que el campo está incorrecto.	1-Seleccionar opción Adicionar.2-Llenar los campos requeridos.3-Aceptar.4-Se selecciona el proyecto a editar.5-Cambia los campos que desea editar.6-Aceptar.7-Al introducir un dato incorrecto alerta al usuario que el dato está incorrecto, mostrando un mensaje "Ya existe un proyecto creado con ese nombre".
EC 1.3 Editar un proyecto con campos en blanco.	Se verifica si se introducen todos los datos correctamente sin quedar ningún campo en blanco.	NA	NA	Al quedar todos los campos vacíos el sistema muestra un mensaje de aviso alertando que no editó ningún campo.	1-Seleccionar opción Adicionar.2-Llenar los campos requeridos.3-Aceptar.4-Se selecciona el proyecto a editar.5-Cambia los campos que desea editar.6-Aceptar.7-Al quedar todos los campos vacíos el sistema muestra un mensaje "No ha actualizado ningún elemento del proyecto".

Nota: las celdas de las tablas contienen V, I, o N/A. V indica válido, I indica inválido y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Para ver el resto de los casos de prueba remitirse a los documentos complementarios, Artefactos, Casos de Prueba.

3.4. Presentación de los resultados de las pruebas

Se evidencia a continuación los resultados de las No Conformidades (NC) clasificadas en significativas y no significativas de las pruebas efectuadas en las tres iteraciones realizadas, donde en la primera iteración se descubrieron un total de 15 NC: 5 errores de validación, 7 errores ortográficos y 3 errores de interfaz. En la segunda iteración fueron encontradas 6 NC: 2 errores de validación y 4 errores ortográficos. Para culminar en la tercera iteración no se encontró ninguna NC, obteniendo resultados satisfactorios. La Fig.21 muestra el comportamiento de la cantidad de NC de ortografía, interfaz y validación en las tres iteraciones realizadas.

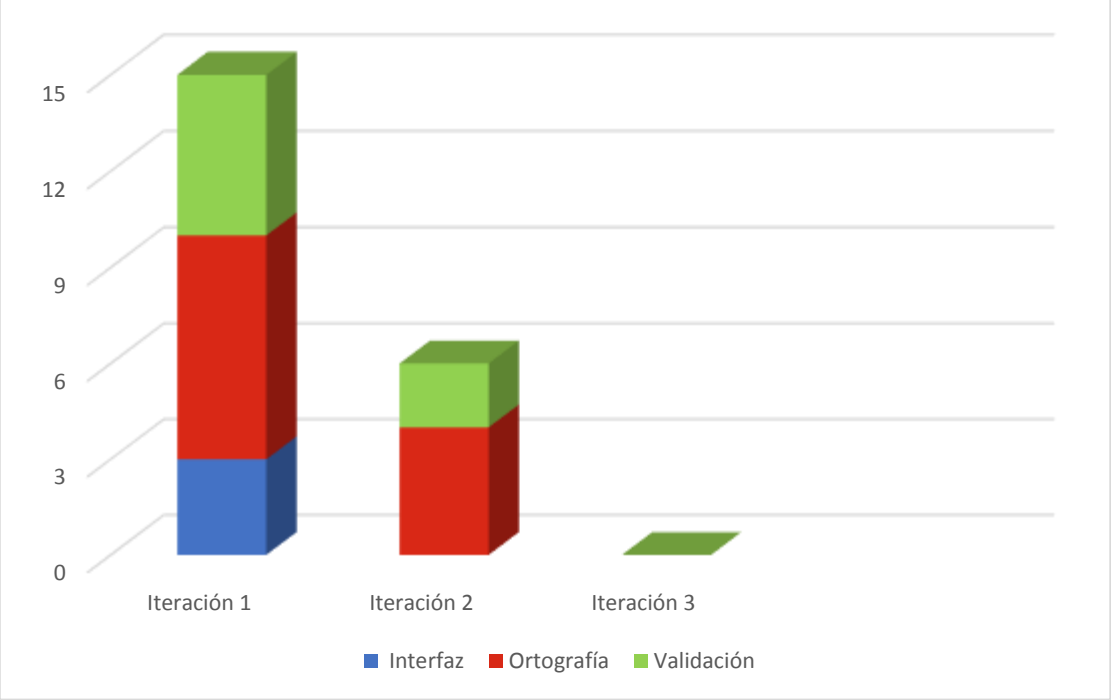


Fig.21 Representación de las clasificaciones de no conformidades encontradas en las iteraciones.

Conclusiones del capítulo

De las 7 HU definidas en el capítulo anterior se logró implementar el 100% de las mismas haciendo uso de PHP 5.3 como lenguaje de programación. Durante el desarrollo del presente capítulo se realizaron 34 TI para el desarrollo de las HU identificadas. Se definieron los estándares de codificación para facilitar una mejor visualización del código. Se realizaron las pruebas funcionales y de aceptación, utilizando el método de caja negra con la técnica partición de equivalencia a través del diseño de casos de prueba, arrojando como resultado 15 NC que fueron solucionadas en tres iteraciones.

Conclusiones generales

La realización del presente trabajo posibilitó cumplir con los objetivos y tareas propuestas para su desarrollo, arribándose las siguientes conclusiones:

- Se creó el propio motor de generación de la herramienta, permitiendo generar datos no sólo de forma aleatoria, sino también mediante listas o archivos externos.
- Se implementó la herramienta *web* para la generación de datos en bases de datos PostgreSQL que permite la población de bases de datos relacionales.
- Se realizaron pruebas funcionales y de aceptación a la aplicación, arrojando resultados satisfactorios.

Recomendaciones

Después de lograr los objetivos que se trazaron al principio del trabajo, se plantean las siguientes recomendaciones:

- Continuar el desarrollo de la herramienta con el objetivo de incorporar otras formas de generar los datos, como a través de consultas *sql* y valores generados sobre otras tablas.
- Incluir la generación de los siguientes tipos de datos: *money, interval, bytea, tsquery y xml*.

Bibliografía

ALCALDE, A. 2012. Los 11 mejores frameworks gratuitos para aplicaciones web. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://elbauldelprogramador.com/los-10-mejores-frameworks-gratis-de-aplicaciones-web/.

ALEGSA, L. 2015. ¿Cuál es la definición de herramienta de modelado? [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.alegsa.com.ar/Dic/herramienta%20de%20modelado.php.

ÁLVAREZ, D.V. 2012. Aplicaciones web vs aplicaciones de escritorio. [en línea]. [Consulta: 25 junio 2015]. Disponible en: http://www.webprogramacion.com/356/blog-informatica-tecnologia/aplicaciones-web-vs-aplicaciones-de-escritorio.aspx.

ÁLVAREZ, S. 2007. Sistemas Gestores de Bases de Datos. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Aplicaciones web vs escritorio. [en línea] [sin fecha]. [Consulta: 25 junio 2015]. Disponible en: http://www.sistemacontablegssecuador.com/sb/index.php/webvsescritorio.

BAHIT, E. 2011. Introducción al desarrollo ágil con scrum. [en línea]. [Consulta: 25 junio 2015]. Disponible en: http://www.desarrolloweb.com/articulos/desarrollo-agil-scrum.html.

COELHO, F. 2014. DataFiller 2.0.0 - generate random data from database schema. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: https://www.cri.ensmp.fr/people/coelho/datafiller.html.

DEBIAN.ORG 2015. Details of package pgadmin3 in squeeze-backports. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: https://packages.debian.org/es/squeeze-backports/pgadmin3.

DESCARGARVISTA.COM 2008a. Data Generator for Oracle 2.3. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://descargarvista.com/sema-2005-generador-de-datos-para-oracle-2.3.zip/1f2bc68c0.

DESCARGARVISTA.COM 2008b. ForSQL generador de datos 1.0. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://descargarvista.com/forsql-generador-de-datos-1.0.zip/408486.

DOWNLOAD.COM 2013. Visual Paradigm for UML. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://descargar.cnet.com/Visual-Paradigm-for-UML/3000-2247_4-42700.html.

DOWNLOADV.COM 2013a. Data Generator for DB2. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://es.downloadv.com/download-EMS-Data-Generator-for-DB2-213257.htm.

DOWNLOADV.COM 2013b. Data Generator para SQL Server 2.2. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://es.downloadv.com/download-EMS-Data-Generator-2005-para-SQL-Server-545200.htm.

EDITORBFB 2011. ¿Qué es un entorno de desarrollo integrado IDE? [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.

EDITORESDECODIGO.COM 2014. Descargar PhpStorm Full IDE de programación web. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html.

EGUILUZ, J. 2014. *Desarrollo web ágil con Symfony2*. S.l.: s.n.

FERNANDES AGRELA, ANA KARINA, MIRANDA BOUZAS y CARLOS MANUEL, 2009. *AAR5162* [en línea]. 2009. S.l.: s.n. Disponible en: http://biblioteca2.ucab.edu.ve/anexos/biblioteca/marc/texto/AAR5162.pdf.

GILES, D. [sin fecha]. Datagenerator. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://dominicgiles.com/datagenerator.html.

GROSSO, A. 2011. Patrones GRASP. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

GUERRERO, L.M.B. 2006. *Arquitectura de software como eje temático de investigación* [en línea]. S.l.: s.n. Disponible en: http://www.unilibre.edu.co/revistaavances/avances-4/r4_art8.pdf.

INGENIERIADESOFTWARE.MEX 2015. Ingeniería de Software. Programación Extrema XP. [en línea]. [Consulta: 24 junio 2015]. Disponible en: http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

KIOSKEA.NET 2015. Lenguajes de programación. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://es.kioskea.net/contents/304-lenguajes-de-programacion.

KIOSKEA.NET 2015. Patrones de diseño. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://es.kioskea.net/contents/224-patrones-de-diseno.

KRALL, C. 2015. ¿Qué es y para qué sirve UML? [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

LAPUENTE, M.J.L. 2013. Bases de datos. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.hipertexto.info/documentos/b_datos.htm.

LASTRA, R.O. 2010. Pruebas alfa y beta. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: https://rfolivares.wordpress.com/2010/09/08/pruebas-alfa-y-beta/.

LÍBER BATALLA, DIEGO FONTANAROSSA, JAVIER GARDERES y ANDRÉS GATTO, 2006. *Gestión de Software*. 2006. S.l.: s.n.

MARCO, B.S. 2013. Expresiones regulares. PHP. [en línea]. [Consulta: 31 mayo 2015]. Disponible en:

http://www.mclibre.org/consultar/php/lecciones/php_expresiones_regulares.html.

MARTÍNEZ, R. 2010. Sobre PostgreSQL. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

MESTRAS, J.P., 2009. *Estructura de las aplicaciones orientadas a objetos. El patrón Modelo-Vista-Controlador (MVC)* [en línea]. 2009. S.l.: s.n. Disponible en: https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

NETBEANS.ORG 2015. Bienvenido a NetBeans. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: https://netbeans.org/index_es.html.

PHP.NET 2015. PHP. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://php.net/manual/es/migration53.new-features.php.

POSTGRESQL 2015. PostgreSQL: Documentation: 8.4: Data Types. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.postgresql.org/docs/8.4/static/datatype.html.

PRESSMAN, R.S. 2011. *Ingeniería del Software. Un Enfoque Práctico. 6ta Edición*. S.l.: s.n.

PRIETO, F., 2009. *Patrones de diseño*. 2009. S.l.: s.n.

PROYECTOSAGILES.ORG [sin fecha]. ¿Cómo gestionar proyectos con Scrum? [en línea]. [Consulta: 25 junio 2015]. Disponible en: http://www.proyectosagiles.org/que-es-scrum.

RUBIO, J.P.M. 2014. Metodologías de desarrollo de software. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://metodologiasdedesarrollodesoftware.blogspot.com/.

SENTIDOWEB.COM 2009. Lista de herramientas para generar datos de prueba. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://sentidoweb.com/2009/01/27/lista-de-herramientas-para-generar-datos-de-prueba.php.

SOSA, I.H. 2013. Tareas de la ingeniería de requisitos. [en línea]. [Consulta: 1 junio 2015]. Disponible en: http://insitutotec.blogspot.com/2013/03/21-tareas-de-la-ingenieria-de-requisitos.html.

SQLMANAGER.NET 2015. Data Generator for PostgreSQL. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://www.sqlmanager.net/products/postgresql/datagenerator.

UPDATESTAR.COM 2015. Data Generator for MySQL 2.3. [en línea]. [Consulta: 31 mayo 2015]. Disponible en: http://ems-data-generator-2005-for-mysql.updatestar.com/es.

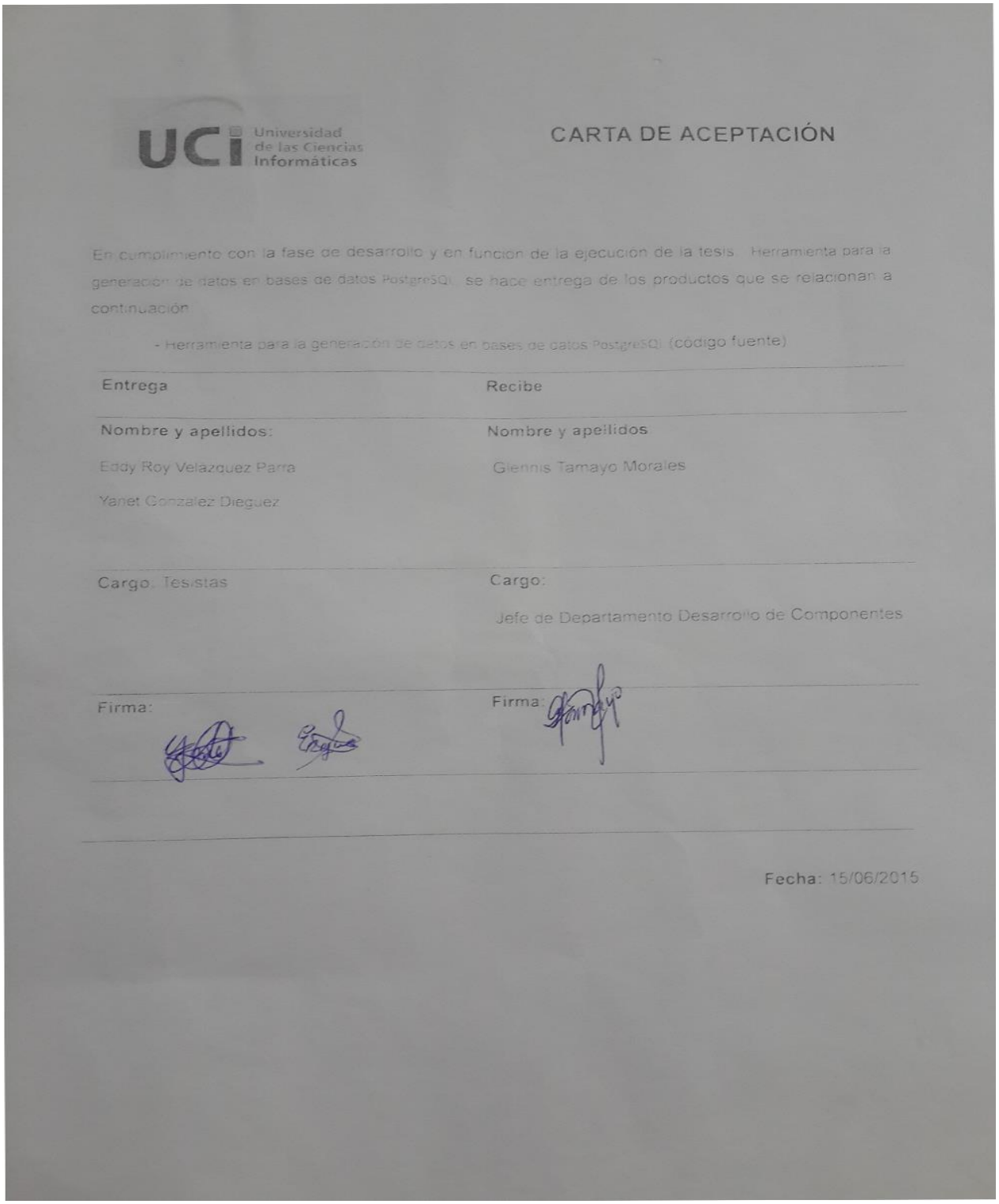
Anexos

Anexo 1: Entrevista

Descripción: guía de preguntas de la entrevista realizada para recopilar información tanto primaria como secundaria para el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL. La entrevista fue realizada al ingeniero: Daymel Bonne Solís.

1. ¿Por qué surge la propuesta de desarrollar una herramienta *web* para la generación de datos en bases de datos PostgreSQL?
2. ¿Qué objetivo se persigue con el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL?
3. ¿Qué ventajas ofrece utilizar una herramienta *web* en vez de *desktop*?
4. ¿Cree útil el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL en la universidad?
5. ¿Qué *framework* considera que se deba utilizar para el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL?
6. ¿Qué lenguaje de programación se utilizará para el desarrollo de la herramienta *web* para la generación de datos en bases de datos PostgreSQL?
7. ¿Sobre qué versión de PostgreSQL resulta más conveniente trabajar?

Anexo 2: Carta de aceptación



UCI Universidad de las Ciencias Informáticas

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución de la tesis "Herramienta para la generación de datos en bases de datos PostgreSQL", se hace entrega de los productos que se relacionan a continuación:

- Herramienta para la generación de datos en bases de datos PostgreSQL (código fuente)

Entrega	Recibe
Nombre y apellidos:	Nombre y apellidos:
Eddy Roy Velazquez Parra	Glennis Tamayo Morales
Yanet Gonzalez Dieguez	
Cargo: Tesistas	Cargo:
	Jefe de Departamento Desarrollo de Componentes
Firma:	Firma:

Fecha: 15/06/2015

Anexo 3: Interfaz principal