UNIVERSIDAD DEL PAPALOAPAN

Campus Loma Bonita

INGENIERÍA EN COMPUTACIÓN

"SISTEMA MIPYME-CAEO PARA LA CREACIÓN DE UN DIRECTORIO DE MIPYMES DE LA CIUDAD DE TUXTEPEC, OAXACA"

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTA
FREDDY ARTURO SÁNCHEZ JIMÉNEZ

ASESOR DE TESIS
M.C. ISAAC MACHORRO CANO

CO-ASESORES DE TESIS
M.C. CARLOS MANUEL COPTO DEL PUERTO
M.C. MÓNICA GUADALUPE SEGURA OZUNA

LOMA BONITA, OAX. 2016

UNIVERSIDAD DEL PAPALOAPAN

INGENIERÍA EN COMPUTACIÓN

LA PRESENTE TESIS TITULADA **"SISTEMA MIPYME-CAEO PARA LA CREACIÓN DE UN DIRECTORIO DE MIPYMES DE LA CIUDAD DE TUXTEPEC, OAXACA"** PRESENTADA POR EL SUSTENTANTE DE LA LICENCIATURA C. **FREDDY ARTURO SÁNCHEZ JIMÉNEZ**, BAJO LA DIRECCIÓN DEL **M.C. ISAAC MACHORRO CANO**, CO-DIRECCIÓN DEL **M.C. CARLOS MANUEL COPTO DEL PUERTO** Y LA **M.C. MÓNICA GUADALUPE SEGURA OZUNA**, HA SIDO ACEPTADA Y REVISADA POR EL COMITÉ EXAMINADOR INDICADO PARA SER DEFENDIDA EN EL EXAMEN PROFESIONAL Y OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN.

M.C. ISAAC MACHORRO CANO
ASESOR

M.C. JOSÉ JULIÁN AGUILAR LÁINEZ
PRESIDENTE

M.C. ARIEL LÓPEZ RODRÍGUEZ
SECRETARIO

DR. EDUARDO SÁNCHEZ SOTO
VOCAL

LOMA BONITA, OAXACA. 2016

Agradecimientos

A mis maestros, que a pesar de todo siempre me ofrecieron su ayuda y consejos.

A mi asesor y co-asesores, que me brindaron su tiempo y apoyo para poder terminar este proyecto.

A mi familia que siempre estuvo ahí para alentarme.

A mi madre, razón y motivo de todo lo que soy ahora, a esa gran mujer le dedico este triunfo.

Gracias

Índice

Índice de figuras

Resumen

Ante la carencia de información clasificada de empresas en la ciudad San Juan Bautista Tuxtepec, Oaxaca identificada por el Cuerpo Académico Estudios Organizaciones (CAEO), surge la necesidad de contar con un directorio de las MiPYMES asentadas particularmente en el primer cuadrante de la ciudad, considerando que es necesario tener una fuente completa, confiable y actualizada. Por tal motivo, se plantea la iniciativa de desarrollar un sistema con el objetivo de crear un directorio de las MiPYMES, basado en un cuestionario que se conforma por una cédula de identificación del negocio (CIN) y una serie de preguntas relacionadas con las temáticas de tecnologías de la información, mercadotecnia, administración y finanzas.

El desarrollo de aplicaciones *Web* en los últimos años se ha beneficiado con la creación de nuevos *Frameworks* creados especialmente para agilizar el desarrollo de las mismas, permitiendo crear páginas *Web* en un tiempo menor y optimizando el código, con el fin de tener un sistema más estructurado y entendible. Actualmente uno de los *Frameworks* de desarrollo de aplicaciones *Web Open Source* más poderosos es *Django*, además es robusto y flexible; además permite desarrollar aplicaciones *Web* de manera rápida, siguiendo un diseño limpio y pragmático.

El aporte principal de esta tesis esla construcción del sistema MiPYME-CAEO desarrollado con el *Framework Django,* que se logró realizando la adaptación de la estructura de la Base de Datos (BD) conforme a los modelos del *Framework,* teniendo como resultados la generación de un directorio de las MiPYMES del primer cuadrante de la ciudad de Tuxtepec, Oaxaca y la obtención de documentos para el análisis de la información por parte de los integrantes del CAEO, de acuerdo al área de tecnologías de la información, mercadotecnia, administración y finanzas de las MiPYMES.

Abstract

Due to the lack of organized data on the small businesses in San Juan Bautista Tuxtepec, Oaxaca, the Cuerpo Academico Estudios Organizaciones (CAEO) identified the need to create a directory of Micro, Small and Medium sized Enterprises (MiPYMEs), starting with the first quadrant of the city, considering it is necessary to have a source of complete, reliable and updated information. This necessity spurred the idea of development of a small business directory Therefore, the initiative to develop a system with the goal of creating a directory of small businesses, based on a questionnaire that is formed by a business identification card (CIN) and a series of questions related to the topics of technology, emerged information, marketing, management and finance.

The development of Web applications in recent years have benefited from the creation of new frameworks, which help to expedite the development process. These frameworks also optimize code, making it more organized and efficient.Today, Django is one of the most powerful, flexible and robust Web development frameworks. It allows for rapid application development using its clean and pragmatic design approach.

The main contribution of this thesis is the construction of MiPYME-CAEO system, developed with Django Framework, which was achieved by adapting the database (BD) structures to the Django model framework, and generating a directory of MSMEs in the first quadrant of the city of Tuxtepec, Oaxaca and then obtaining documents for the analysis of information by CAEO organization, pertaining to information technology, marketing, management and finance areas of small businesses.

Introducción

En la actualidad el papel que representan las empresas en México es visible en la creación de empleos, crecimiento y desarrollo, debido a que existen aproximadamente 4 millones 15 mil unidades empresariales, de las cuales 99.8% son Micro, Pequeñas y Medianas Empresas (MiPYMES) y generan el 52% del Producto Interno Bruto (PIB) y el 72% del empleo en el país [1]. Por otra parte la región de la Cuenca del Papaloapan se caracteriza por su amplia actividad económica ya que es un punto de convergencia de las actividades económicas de los estados de Oaxaca, Veracruz y Puebla, además posee una importante actividad en los sectores primario, secundario y terciario. En esta región se encuentra la ciudad de San Juan Bautista Tuxtepec, Oaxaca, donde la actividad económica predominante y de mayor crecimiento en la ciudad es el sector comercial, tal es su importancia que se considera que aproximadamente el 80% de la población se dedica a ella, lo cual la convierte en la ciudad más importante de la región de la Cuenca del Papaloapan en materia comercial. Este sector mantiene relaciones comerciales tanto con ciudades dentro del mismo estado como con ciudades de otros estados de la república como Veracruz, Puebla, Distrito Federal, Guadalajara y Monterrey entre otros, además debido a su ubicación geográfica es un punto estratégico para actividades comerciales de las comunidades y lugares circunvecinos de la región [2].

Sin embargo actualmente organismos como la Cámara Nacional de Comercio (CANACO) y la Secretaría de Turismo y Desarrollo Económico (STyDE) no cuentan con un directorio actualizado y completo de las MiPYMES asentadas en la ciudad, que pueda ser utilizado como una fuente de información. Por tal motivo, en esta tesis se presenta el desarrollo del sistema MiPYME-CAEO, con el cual se logró obtener un directorio de las MiPYMES de la ciudad de San Juan Bautista Tuxtepec, Oaxaca, basado en un instrumento de recolección de datos desarrollado por el Cuerpo Académico Estudios Organizacionales (CAEO) de la Universidad del Papaloapan (UNPA).

El sistema MiPYME-CAEO fue desarrollado en *Django*, que es un *Framework* de desarrollo *Web*, de alto nivel y de código abierto escrito en *Python*, siendo este uno

de los lenguajes de programación más poderosos que existen, además de ser robusto y flexible. *Django* permite desarrollar aplicaciones *Web* de manera muy rápida, siguiendo un diseño limpio y pragmático debido a su sistema de manejo MTV (*Model-Template-View*). MTV nos brinda características tales como un rápido y eficaz mapeo de URLs y la posibilidad de gestionar los *templates* de manera que se puedan ahorrar cientos de líneas de código, e inclusive se puede utilizar herencia entre *templates* [3] [4].

Para la Base de Datos (BD) se utilizó el Sistema Gestor de Bases de Datos (SGBD) MySQL. MySQL es un SGBD creado para brindar mejor desempeño en las BD enfocadas a aplicaciones *Web* debido a la gran cantidad de consultas que se generan a cada momento en estas aplicaciones. MySQL es una herramienta flexible, escalable, pudiendo trabajar en servidores con pocas características. Además MySQL se integra de manera eficiente al *Framework Django*, permitiendo a *Django* realizar consultas por medio de los modelos. *Django* se encarga de gestionar las conexiones con la BD ahorrándole con esto al usuario tener que lidiar con los accesos y los diversos métodos de seguridad que se requieren al momento de ingresar a la BD [5].

La estructura de la tesis se representa de la siguiente manera: en el capítulo 1, se describe el planteamiento del problema, los antecedentes, el estado del arte y se presenta la propuesta de solución. En el capítulo 2, se describen herramientas para el desarrollo de aplicaciones Web, así como sus ventajas y desventajas. En el capítulo 3, se abordan las metodologías ágiles más utilizadas para el desarrollo de software. En el capítulo 4, se describen las fases para el desarrollo del sistema MiPYME-CAEO y se indican los pasos para la generación del directorio de las MiPYMES de la ciudad de Tuxtepec, Oaxaca. Finalmente en el capítulo 5, se presentan las conclusiones generales, las contribuciones de esta tesis y el trabajo a fututo.

Esta tesis fue apoyada y financiada por el Programa para el Desarrollo Profesional Docente (PRODEP) en su convocatoria de fortalecimiento de cuerpos académicos a través del CAEO de la UNPA.

Capítulo 1. Motivación

Introducción

En este capítulo se mencionan los antecedentes de desarrollo de software, el planteamiento del problema, los objetivos, la hipótesis y justificación para el desarrollo del sistema MiPYME-CAEO. Además se presentan los trabajos relacionados y la propuesta de solución para generar un directorio de las MiPYMES de la ciudad de Tuxtepec, Oaxaca.

1.1 Antecedentes

Los constantes avances tecnológicos y las necesidades de actualizar los sistemas informáticos de forma que se mantengan a la vanguardia hacen del desarrollo de software una parte muy importante al momento de crear un nuevo sistema. De manera general se requiere que el sistema cumpla con los objetivos previstos y sea confiable al momento de usarlo, pero también se requiere de un modelo estándar de desarrollo que permita la fácil comprensión del sistema para su mantenimiento y que además se tenga la posibilidad de actualizarlo según se requiera. La llegada del Internet en el ámbito laboral ha expandido horizontes que antes las pequeñas empresas no podían vislumbrar, ahora expandir los negocios es posible utilizando las redes sociales o creando una página para la empresa, con esto la empresa abarca un mayor sector de la población hacia la cual se enfocan sus productos y/o servicios. Para las MiPYMES esto también implica posibles expansiones de personal y ubicaciones, por lo que el uso de las Tecnologías de la Información (TI) cada vez más será parte primordial para el funcionamiento de la organización. A su vez las TI ayudan a las empresas marcando tendencias en cuanto a las características que deben cumplir los servicios y productos, estos datos son obtenidos mediante encuestas en línea, o por medio de estudios realizados por otras empresas y que han puesto públicos los datos obtenidos [6].

Por otra parte, la innovación tecnológica ha sido una de las principales razones

por las cuales exista mayor competencia en el mercado, marcando estándares de calidad en cuanto al proceso de desarrollo de productos o la calidad de los materiales utilizados para la creación del mismo. Estas características benefician directamente al usuario final debido a la gran calidad en los productos y servicios que obtienen de las diversas empresas en el mercado y a su vez a las empresas ya que ofrecen productos y servicios de calidad que cumplen con características óptimas. Cada empresario debe saber lidiar con las tecnologías de la información y comunicación (TIC) y en base a ellas generar estrategias que hagan prevalecer sus empresas [7].

Las TIC han permitido a las empresas ahorrar grandes cantidades de dinero ya que muchas de ellas agilizan las funciones necesarias para la empresa, otras son de libre acceso y crean una buena conexión entre sus clientes y proveedores. Por lo tanto, las empresas tienen mayores posibilidades de mejorar sin la necesidad de hacer grandes inversiones en el desarrollo de software o en investigaciones de mercado [8].

1.2 Planteamiento del problema

El municipio de San Juan Bautista Tuxtepec, Oaxaca, fue fundado en el año de 1811 y decretado como municipio por la cámara de comercio local el 15 de marzo de 1825. Se localiza en la región de la Cuenca del Papaloapan y es además rodeada por las aguas del río Papaloapan, emblema de la ciudad, colinda al norte con el estado de Veracruz y el municipio de San Miguel Soyaltepec, al sur con los municipios de Santiago Jocotepec y Loma Bonita, al poniente con los municipios de Santa María Jacatepec, San Lucas Ojitlán y San José Chiltepec y al oriente con el municipio de Loma Bonita. Se ubica como la segunda ciudad más poblada del estado de Oaxaca, su economía está basada en diferentes sectores como son: agricultura, ganadería, pesca, servicios, industria, turismo y comercio principalmente [2]. En el año 2012 se realizaron algunos esfuerzos por parte de la CANACO y STyDE de la ciudad para llevar a cabo un registro de las MiPYMES de las principales avenidas para conocer la información del sector comercial y de servicios de la ciudad, sin embargo la información obtenida fue insuficiente y solo se obtuvieron datos generales de algunas empresas.

Por su parte el CAEO propuso generar un directorio de las MiPYMES de la ciudad de San Juan Bautista Tuxtepec, Oaxaca, considerando que es necesario tener una fuente de información completa, confiable y actualizada. Por tal motivo, se desarrolló el

sistema MiPYME-CAEO con el objetivo de crear un directorio de las MiPYMES asentadas en la ciudad, basado en un cuestionario que se conforma por una cédula de identificación del negocio (CIN) y una serie de preguntas relacionadas con las temáticas de tecnologías de la información, mercadotecnia, administración y finanzas.

1.3 Objetivo general

Desarrollar el sistema MiPYME-CAEO en el *Framework Django* que permita generar un directorio de las MiPYMES de la ciudad de Tuxtepec, Oaxaca.

1.4 Objetivos específicos

1. Estudiar, analizar y comprender las secciones del cuestionario elaborado por el CAEO y aplicado a las MiPYMES de la ciudad de Tuxtepec, Oax.
2. Identificar los tipos de preguntas y posible(s) respuesta(s) que conforman las diversas secciones del cuestionario.
3. Estudiar, analizar y comprender la metodología ágil *Adaptative Software Development* (ASD).
4. Definir la estructura de la base de datos conforme a los modelos del *Framework Django*.
5. Investigar y estudiar las funciones y principales aspectos del *Framework Django*.
6. Diseñar los templates correspondientes para cada sección y anexos que conformarán el sistema MiPYME-CAEO.
7. Realizar las pruebas y puesta en marcha del sistema MiPYME-CAEO.
8. Generar el directorio de las MiPYMES de la ciudad de Tuxtepec, Oax.

1.5 Hipótesis

Mediante el desarrollo del sistema MiPYME-CAEO utilizando el *Framework Django* será posible generar un directorio de las MiPYMES de la ciudad de San Juan Bautista Tuxtepec, Oaxaca, que permita obtener información completa, confiable y actualizada de las MiPYMES del sector comercial y servicios.

1.6 Justificación

La motivación de desarrollar el sistema MiPYME-CAEO es solucionar una problemática necesaria y real en el área de las ciencias empresariales tal y como la presenta el CAEO, aplicando los conocimientos adquiridos en la Ingeniería en Computación en otra disciplina, generando un nuevo aprendizaje al utilizar el Framework Django que cuenta con características que nos permite un rápido desarrollo de sistemas, como lo es el mapeo objeto-relacional (ORM), técnica que permite la conversión de datos entre un lenguaje de programación orientado a objetos (OOP) y una BD relacional como MySQL, ya que con esta técnica se conservan características de la OOP como la herencia y el polimorfismo que hacen de la gestión de los datos más fácil y eficiente. Además, otra motivación es porque al desarrollar un sistema para crear un directorio de las MiPYMES asentadas en la ciudad, se tendrá una fuente de información completa, confiable y actualizada que sea utilizada como una herramienta para conocer la situación actual y plantear estrategias que contribuyan al desarrollo económico del sector comercial y de servicios.

Dentro de los principales beneficiarios al llevar a cabo esta propuesta es primeramente el CAEO ya que contarán con información relevante de las MiPYMES para poder clasificarlas de acuerdo a los sectores industrial, comercial y servicios, además podría establecer un diagnóstico de las problemáticas y perspectivas de las mismas, teniendo una importante base para la creación de un directorio de todo el municipio y de otros municipios circunvecinos, además de poder utilizar la información del directorio para investigaciones futuras; otros organismos beneficiados serían la CANACO y la STyDE ya que podrán tener acceso a la información del directorio para poder así contar con información completa y actualizada de las MiPYMES de la ciudad, promoviendo capacitación, programas de financiamiento y gestión empresarial para el desarrollo económico de la región del Papaloapan; finalmente todas las MiPYMES de la ciudad de Tuxtepec podrían ser beneficiadas ya que podrían tener información de fuentes de financiamiento, cursos de capacitación y apoyo tanto del CAEO como de la CANACO y de la STyDE.

1.7 Trabajos relacionados

En [9] se presenta la creación de un directorio con empresas de servicios documentales de España, con el fin de tener un registro de este tipos de empresas, poder tener acceso a ellas para posibles contrataciones y también para que los recién egresados universitarios tengan un directorio en donde poder ubicar las empresas y saber cuáles de ellas están solicitando personal para el área de su especialidad.

Además en [10], se presenta el resultado de un estudio sobre diversas MiPYMES de Colombia, en donde se consultó a todas ellas sobre el uso y disponibilidad que podrían tener sobre ocho diferentes herramientas TIC en un ambiente Web y el rendimiento aportado por ellas, así como la rentabilidad y alcances que podrían obtenerse en el mercado. El estudio demostró que las TIC claramente son un factor importante que influye en el desarrollo de las MiPYMES.

Por otra parte en [11] se menciona que se creó un directorio tomando como referencia empresas españolas de reciente creación que desarrollan sus actividades en el área de biotecnología aplicada a la salud humana. Esto con el fin de fomentar la investigación científico-tecnológica entre los diferentes agentes del sector farmacéutico español.

Asímismo en [12], se menciona que en base al gran aumento de comercio electrónico entre las pequeñas y medianas empresas (PYMES), se creó un directorio de empresas en Hermosillo, Sonora donde a cada empresa se le realizó un estudio para saber el impacto que tendría su empresa al ingresar en el comercio electrónico. Esto con el fin de poder ampliar el margen de venta de las PYMES, dando oportunidad de comercializar sus servicios utilizando herramientas informáticas.

En [13] se explica como con el avance de la tecnología y las nuevas necesidades que presenta una PYME requieren que se gestione la empresa de una manera más automatizada y que ofrezca mejor manejo de la información, en este caso utilizar bases de datos en PYMES es ya una necesidad, debido muchas veces a la gran cantidad de datos que se requiere manipular y a su vez la protección de los mismos, ya que son causa de filtración de información delicada y/o privada.

Por otro lado en [14], se presenta un análisis de la relación existente entre los métodos y estrategias empleadas por los dueños de las pequeñas empresas al momento de adaptarse al medio ambiente y el uso de las tecnologías de información y

comunicaciones (TIC). Utilizando el software estadístico SPSS con el cual se obtuvieron resultados que demuestran que no existe una relación específica entre los métodos utilizados por los dueños de las empresas y las TIC.

Además en [15], se explica el por qué de la necesidad de software flexible en PYMES, permitiendo con ello poder editar, mejorar y expandir las propiedades del sistema, con el fin de mantener un buen rendimiento. Todo esto utilizando la mínima cantidad de recursos e inclusive ofreciendo la oportunidad de disminuir los gastos.

Asímismo en [16], se comenta sobre la implementación de un sistema de planificación de recursos empresariales (ERP, por sus siglas en inglés, *Enterprise Resource Planning*) como un servicio en la nube (SaaS, por sus siglas en inglés *Software as a Service*), y la forma en que permite a las PYMES un mejor manejo y gestión de sus sistemas basados en los aspectos más importantes que propician un desarrollo más productivo de la empresa.

En [17] se analiza el comercio electrónico como factor competitivo para MiPYMES en el sector comercial de Durango, con el fin de orientar a las MiPYMES sobre las bondades que conllevan implementar el comercio electrónico para obtener una ventaja competitiva en el mercado y fomentar su desarrollo.

Por otro lado en [18] se presenta a BonsaiLIMS, un Sistema Gestor de Información de Laboratorios creado utilizando el *Framework* de *python, Django*. Debido a la necesidad individual de cada laboratorio, requieren un LIMS específico dependiendo del tipo de información que se maneje, lo cual es difícil de implementar. BonsaiLIMS está implementada de tal manera que es fácil agregar módulos y utilizar los ya existentes para adaptarlos a las necesidades de los laboratorios, lo que hace de BonsaiLIMS una herramienta completa, de fácil manejo y fácil implementación en cualquier dispositivo con un ambiente *python*.

Por otra parte en [19] se propone a WSSBI, un sistema de soporte Web para la Inteligencia Empresarial (*BI, Business Intelligence*), la cual es un conjunto de estrategias generadas por el análisis de los datos existentes en una organización o empresa. Estas estrategias permiten obtener mejores resultados a las empresas basándose en sus propios datos generados y facilitan la toma de decisiones con respecto a diversas cuestiones relacionadas con ámbitos de trabajo, relaciones laborales y datos financieros. WSSBI es una herramienta en línea que reúne varias de las estrategias más utilizadas por las diversas MiPYMES. Los detalles más importantes de la herramienta

son que se encuentra alojada en un host en línea y se tiene acceso desde cualquier dispositivo con Internet, no requiere de conocimiento sobre BI y es posible crear modelos personalizados de estrategias para empresas con operaciones específicas.

Por otra lado en [20] se presenta una metodología de desarrollo de bases de datos que toma en cuenta las limitaciones y características de las MiPYMES, tomando particularidades de las metodologías ágiles que se adecuan a este tipo de empresas, como lo son: el uso de algunos artefactos, escasos roles, pequeños grupos de trabajo, la participación constante del cliente y la explicación detallada de las diversas actividades.

1.8 Propuesta de solución

Basados en la problemática descrita anteriormente y a la complejidad de posibles errores al elaborar los resultados y al realizar cálculos estadísticos, se propone el desarrollo del sistema MiPYME-CAEO, que plantea un sistema automatizado que realice un directorio de MiPYMES, además de los procesos y cálculos requeridos para el diagnóstico y análisis de la situación actual de las MiPYMES, tales como obtener el porcentaje de establecimientos que representan, así como el grado de estudios que actualmente tienen, tanto los dueños de las empresas como los trabajadores, entre otros aspectos. Dicha información servirá para la investigación y generación de conocimiento para el CAEO y como fuente confiable para diversas organizaciones públicas; además la misma información será usada para la capacitación de las empresas. En la Figura 1.1 se muestra la propuesta de solución para el desarrollo del sistema MiPYME-CAEO, que se pretende desarrollar a través de un sistema *Web* creado con el *Framework Django*, la cual es una de las herramientas de desarrollo más utilizadas actualmente en la industria del desarrollo *Web*, gracias a su facilidad de implementación y aprendizaje. Es una herramienta de libre acceso, lo que ha generado una gran comunidad de desarrollo encargada de dar soporte a los *bugs* y errores que llegan a surgir en el sistema, y también un gran apoyo hacia los usuarios que empiezan a utilizar el *Framework*.

Figura 1.1 Propuesta para el desarrollo del sistema MiPYME-CAEO

Dentro de los alcances esperados en esta propuesta de solución, fundamentalmente se pretende desarrollar el sistema MiPYME-CAEO en el *Framework Django* en combinación con el SGBD MySQL en un equipo con sistema operativo Xubuntu de GNU/Linux utilizando el servidor Apache; todo esto basado en un cuestionario elaborado por el CAEO.

Además el sistema MiPYME-CAEO realizará las operaciones básicas como son el alta de un negocio, modificación de datos del negocio, consulta de negocios y eliminación de un negocio según se requiera. Asímismo se generarán reportes en formas de tablas de todas las secciones y partes que integran el cuestionario para que los integrantes del CAEO visualicen los resultados.

Capítulo 2. Herramientas de desarrollo Web

Introducción

Este capítulo tiene como objetivo describir herramientas de desarrollo, tales como *Frameworks* de desarrollo *Web*, sistemas gestores de bases de datos y servidores Web. Indicando sus principales características, ventajas, desventajas y la selección de las herramientas de desarrollo a utilizar en el sistema MiPYME-CAEO.

2.1 Frameworks de desarrollo Web

Un *Framework* es un conjunto de conceptos estándares, criterios y prácticas enfocados a la resolución de objetivos que sigan un patrón o esquema similar [21]. En el desarrollo *Web* un *Framework* es un conjunto de herramientas, módulos y configuraciones dedicadas a la creación de una aplicación *Web*, tomando como referencia las necesidades comunes que los desarrolladores deben enfrentar al momento de crear una aplicación, y que por ende, se enfoca al uso cómodo de la herramienta y a su vez al desarrollo fluido que agilizan el trabajo del desarrollador. De esta manera el desarrollador se centra en las problemáticas específicas que deban constituir la aplicación [22].

El desarrollo de aplicaciones Web en los últimos años, se ha beneficiado con la creación de nuevos *Frameworks* en focados en agilizar el desarrollo de las mismas, permitiendo crear ahora páginas Web en un tiempo menor al que se empleaba con anterioridad y optimizando el código, esto último con el fin de tener un sistema más estructurado y entendible para los desarrolladores. Además existen otros Frameworks de desarrollo de aplicaciones Web *Open Source* tales como *Rails, Kohana, Spring, Zend Framework, Yii, Pylos, Catalyst, Symfony* y *TurboGears*. A continuación se describen los cuatro *Frameworks* para el desarrollo de aplicaciones Web más utilizados.

2.1.1 Spring

Spring es un *Framework* basado en Java que proporciona un modelo de programación completo para aplicaciones empresariales en cualquier plataforma de desarrollo. Su principal ventaja es su enfoque de infraestructura a nivel de aplicación, esto es que se encarga de todas las configuraciones pertinentes para que la aplicación *Web* sea probada y desarrollada de manera más rápida y eficiente. Con esta ventaja los desarrolladores solo se deben enfocar en la lógica que se debe implementar para que la aplicación realice su propósito principal, también conocida como lógica de negocios a nivel aplicación. *Spring* está pensado para poder desarrollar desde aplicaciones empresariales hasta aplicaciones *Web;* solo es necesaria la implementación de ciertos módulos, los cuales son [23]:

- *spring-web*. Este módulo agrega paquetes básicos de integración *Web* como la carga de archivos desde diferentes clientes.
- *spring-webmvc*. Módulo que agrega las funcionalidades MVC (Model-View-Controller, Modelo-Vista-Controlador) y todos los servicios Web necesarios para trabajar con *spring-webmvc-portlet.*

Spring implementa el uso de POJO (*Plain Old Java Object*) que permite que los desarrolladores no tengan que utilizar las APIs de Java necesarias para la creación de aplicaciones empresariales (EJB, *Enterprise Java Beans*), si no que en vez de esto se utiliza un *Servlet* como Tomcat. Los EJB proveen los medios de comunicación entre la aplicación y el servidor, como transacciones, concurrencias, eventos y seguridad [24].
Para mantener la modularidad, *Spring* utiliza el paradigma de Programación Orientada a Aspectos (POA), que permite la generación de un código más limpio y estandarizado por medio de módulos representados por clases que realizan funciones específicas las cuales a su vez son accesibles desde otras clases. La principal ventaja de la POA es que permite mantener la funcionalidad de una clase como tal y por otro lado implementar aquellas tareas o funciones que no tienen relación directa con el objeto, más sin embargo son necesarias para la resolución de conflictos y objetivos de la clase [25].

2.1.2 Zend Framework [ZF]

Zend es un *Framework* de desarrollo *Web* y servicios, escrito bajo licencias *Open Source*, esto ha facilitado que Zend tenga muchos patrocinadores que a su vez agregan muchas funcionalidades al *Framework*. Zend utiliza todas las bondades y características nuevas que ofrece PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) agregando el uso de código orientado a objetos que facilita de mejor manera la gestión de información y estructuras de datos. ZF utiliza una arquitectura débilmente acoplada para sus componentes lo cual permite que cada componente sea ocupado por separado sin la necesidad de hacer referencia hacia otros componentes. Este tipo de diseño se denomina como "*use-at-will*" (uso a voluntad) [26].

ZF implementa la metodología de desarrollo Modelo–Vista–Controlador (MVC), tal y como se observa en la Figura 2.1. MVC fue concebido para facilitar el desarrollo de aplicaciones *Web* separando en módulos que se encargan de realizar las diversas tareas que conllevan el sistema.

Por un lado se requiere de la representación estructural para el manejo de información, denominada como Modelos, que facilite el acceso de los datos correspondientes a un objeto determinado y que se encuentra en una base de datos, una Vista es una representación visual de la información obtenida por los modelos y que es apreciada por los usuarios mediante un explorador *Web*, y el Controlador es el medio de comunicación entre los modelos y las vistas, este se encarga de recibir y re-direccionar las peticiones realizadas por el cliente y a su vez obtener y estructurar la información de manera que pueda ser visualizada de forma simple y útil para el usuario, una vez obtenidos los resultados deseados el mismo controlador envía la información de vuelta al usuario mediante una vista para que pueda ser analizada [27].

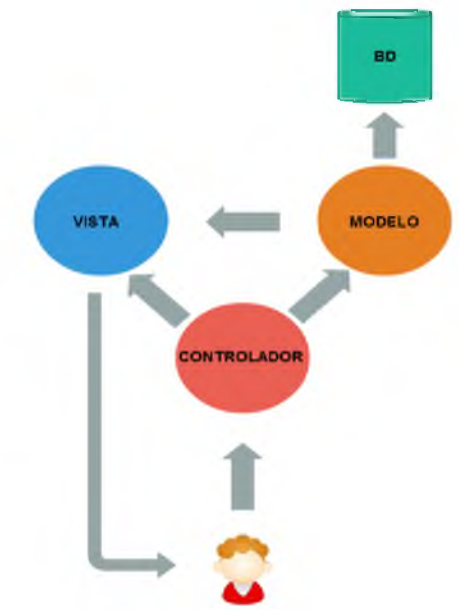


Figura 2.1 Esquema de funcionamiento de MVC. FUENTE: [27]

Además Zend permite vincular el código con herramientas de pruebas como *PHPUnit* y *Travis CI* para depuración y corrección de código. *PHPUnit* es un *Framework* de pruebas unitarias con el propósito de verificar el funcionamiento correcto de las aplicaciones PHP en base a *assertions* (afirmaciones), que consisten en crear un ambiente controlado para la prueba de funciones con el propósito de verificar su funcionamiento. Cada *assertion* se ejecuta sobre funciones aisladas, y se asignan valores que en teoría deban llevar al desarrollo correcto de la función, y a su vez se trata de ingresar parámetros que hagan fallar la función, de esta manera se crean validaciones necesarias para tener una función robusta [28]. Por otra parte, *Travis CI* es un sistema distribuido de generación e integración continua libre, que al igual que *PHPUnit* permite generar rutinas para verificar el código, con la gran ventaja de poder aplicarlas directamente a los repositorios *Github* públicos que pertenezcan a la aplicación. Se crean varias máquinas virtuales para poder configurarlas de tal manera que se verifique la aplicación sin necesidad de tenerla instalada de forma local [29].

2.1.3 Rails

Rails es un *Framework* multiplataforma para desarrollo de aplicaciones *Web* que sigue la arquitectura MVC como metodología de desarrollo. Rails hace referencia sobre

supuestas tareas que los desarrolladores requieren y crea sub-rutinas que satisfagan esas necesidades y faciliten la creación y desarrollo de aplicaciones. Se basa en dos ideologías principales:

1. *DRY ("Don't Repeat Yourself)"*, que promueve la reutilización del código ya existente con el fin de minimizar la escritura en la aplicación.
2. *"Convention Over Configuration"*. Rails hace suposiciones sobre las necesidades que tienen los desarrolladores y realiza pre-configuraciones que agilizan el desarrollo de las aplicaciones *Web*.

Rails ofrece módulos pre-establecidos y que se enfocan en realizar rutinas que son las más requeridas por los desarrolladores, como *Action Mailer*, un módulo para enviar correos electrónicos y *Active Resource* para crear recursos REST (*Representational State Transfer*, Transferencia de Estado Representacional) [30]. Además incluye bibliotecas de *Javascrip* que contienen mejoras hacia las interfaces de usuario. Este *Framework* hace uso de los *plugins* denominados *Gemas,* las cuales son librerías de Ruby que se agregan a las aplicaciones Rails y que realizan tareas comunes como funciones para *login* y *logout* de usuarios, verificación de permisos así como agregar funciones estándares que se encargan de realizar funciones que normalmente los desarrolladores harían por su propia cuenta, pero que gracias a la gran comunidad de Ruby existen más de 100,000 *Gemas* disponibles para su libre uso [31] [32]. Rails sigue un diseño persistente, asumiendo que hay un modo seguro para el diseño de aplicaciones *Web*, entonces Rails empieza a hacer configuraciones conforme a esos lineamientos. Rails no limita al programador a utilizar su sistema de desarrollo, pero se recomienda que para obtener una buena experiencia de desarrollo se sigan las configuraciones que ofrece el *Framework*. Rails acepta el uso de diversas herramientas para ser desplegado en un ambiente de servidor de producción, tales como *Phusion Passenger*, que es un módulo de rails que funciona tanto para Apache como para Nginx y proporciona una interfaz más amigable y hace de la gestión de múltiples páginas *Web* en el mismo servidor más sencilla [33] [34]. Además la posibilidad de utilizar los sistemas gestores de bases de datos más populares en el ambiente de la programación tales como SQLite, MySQL y PostgreSQL. Rails utiliza Ruby, que es un lenguaje de programación creado por *Yukihiro "Matz" Matsumoto*, que se encargó de recopilar las mejores características y configuraciones de otros lenguajes de programación y

unificarlos para crear un lenguaje estable y robusto. Estas características hacen que Ruby sea un lenguaje complicado de comprender y mantener, por lo que se requiere tener experiencia sobre conceptos de programación [35].

2.1.4 Django

Django es un *Framework Web* de alto nivel basado en el lenguaje de programación Python que permite el rápido desarrollo, ya que implementa de manera automática funcionalidades que cada aplicación *Web* usaría por default. Estas herramientas son utilizadas justo después de inicializar un nuevo proyecto y fueron implementadas de manera automática precisamente para evitar que el desarrollador las tenga que implementar por su propia cuenta, esto agiliza la creación de la aplicación, teniendo solo que priorizar los objetivos principales del proyecto. A diferencia de otros *Frameworks* que limitan a utilizar sus métodos de seguridad y otros aspectos, Django permite a los desarrolladores crear sus propias sub-rutinas de desarrollo y la posibilidad de desactivar todos los paquetes pre-instalados en el proyecto para agilizar el proceso de carga del sistema [36].

El *Framework* de desarrollo de aplicaciones *Web* que se utilizó en este trabajo fue Django, ya que además de sus características y ventajas es un *Framework* de desarrollo *Web* multiplataforma de alto nivel y de código abierto escrito en Python. Además Django ha sido uno de los *Frameworks* más elegidos durante la última década gracias a sus cinco particulares características que a continuación se describen:

1. **Robusto**.- Django ofrece por *default* opciones de seguridad tales como protección de ataques XSS (*Cross Site Scripting*), CSRF (*Cross Site Request Forgery*) y *SQL Injection*, que son de las brechas de seguridad más comunes en el desarrollo de aplicaciones *Web*. También se implementan medidas de seguridad como SSL/HTTPS mediante módulos que contiene Django.
2. **Flexible**.- Debido a su flexibilidad y agilidad se crean o re-crean aplicaciones *Web* de manera muy rápida. Esto gracias a la gran cantidad de herramientas ya prediseñadas para realizar las tareas más comúnmente necesitadas y utilizadas por los desarrolladores, desde un módulo de paginación, hasta un módulo de administración.

3. **Fácil de aprender y utilizar**.- Al estar escrito en Python, uno de los lenguajes de programación más potentes y utilizados, Django hereda todas las características y bondades de Python, como la facilidad de comprender el lenguaje y estructurarlo de manera sencilla. Python también es reconocido por aceptar múltiples paradigmas de programación como POO, POA, *Procedural Programming* (Programación de Procedimientos), entre otros [37], característica que beneficia al desarrollador ya que no requiere apegarse a un paradigma en específico como es el caso de Spring.
4. **Reduce tiempo**.- Django fue creado pensando en el rápido desarrollo y actualización de información para las aplicaciones *Web*, que en su momento llevaban un largo tiempo en ser desplegadas en un ambiente de producción debido a que cada vez que se requería realizar una aplicación *Web* esta debía ser escrita desde cero, desde conexiones a Bases de Datos hasta la configuración del servidor [38].
5. **Es libre**.- Django es una herramienta *Open Source* y se tiene acceso directamente al código tanto para modificación y optimización del mismo según las necesidades de los programadores.

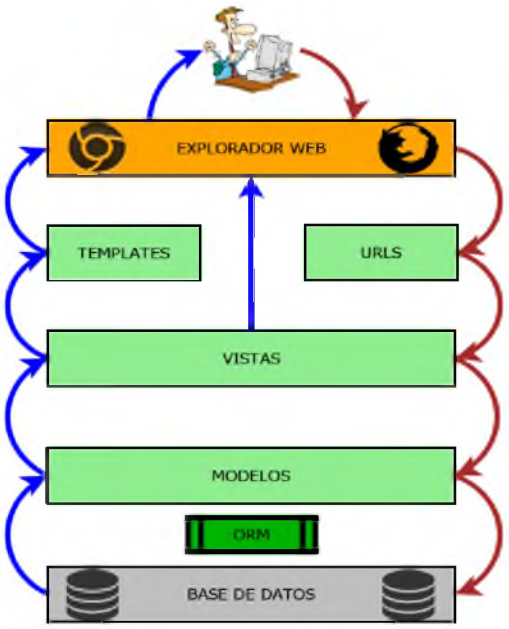


Figura 2.2 Esquema de funcionamiento de MTV. FUENTE: [39]

Django funciona bajo una arquitectura similar a MVC, conocida como Modelo–*Template–View* (MTV), en la Figura 2.2 se muestra su funcionamiento. Además los

modelos al igual que MVC representan un medio de comunicación entre el SGBD y el *Framework*, también llamada capa de acceso de información, en la Figura 2.3 se observa la representación de un modelo en el Framework Django. En la capa de acceso se generan consultas utilizando el sistema de Mapeo–Objeto–Relacional (ORM), una técnica que permite la conversión de uno o más objetos (modelos) en tablas pertenecientes a una BD y a su vez por medio de los mismos modelos obtener la información requerida de esas tablas sin realizar consultas en lenguaje SQL.

```python
from django.db import models


class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')


class Choice(models.Model):
    question = models.ForeignKey(Question)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Figura 2.3 Modelos de Django. FUENTE: [39]

Por otra parte, los *templates* estructuran la información hacia el navegador *Web* utilizado por el usuario y se encargan de definir como será visualizada de manera que el usuario la pueda analizar y comprender sin problema alguno. Los *templates* de Django representan una de las características más importantes de Django gracias a la propiedad de herencia que tienen, puesto que un *template* hereda toda su estructura hacia otro, esto comúnmente es utilizado para crear *templates* "base" que reutilizan la estructura de un template y la agregan en otro. En la Figura 2.4, se muestra un ejemplo de la sintaxis de un *template* en Django.

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Figura 2.4 Ejemplo de código para un *template* de Django. FUENTE: [38]

Además en el caso de Django los *views* representan la capa de lógica de negocios que trabaja como intermediario entre los modelos y *templates*. En los *views* se realizan todas las funciones que obtendrán la información requerida de los modelos y que más adelante será enviada a los *templates* para mostrarla al usuario. Cabe mencionar que el controlador en este caso es representado por el mismo *Framework* Django, que gestiona las peticiones por medio de un archivo llamado *urls.py*; este es un archivo python que guarda en una lista todas las *url* (*uniform resource locator*, localizador de recursos uniforme) que son accesadas en el sistema y sus re-direccionamientos, tal y como se muestra en la Figura 2.5. Por medio de este archivo es que Django gestiona todas las peticiones que se hagan hacia el sistema y es la razón por la cual el Controlador no se encuentra implícito dentro de la estructura del MTV [39].

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^polls/', include('polls.urls', namespace="polls")),
    url(r'^admin/', include(admin.site.urls)),
]
```

Figura 2.5 Código para el archivo urls.py que ejerce la función de controlador en Django. FUENTE: [38]

Otro de los grandes atributos de Django es su interfaz de administración automatizada que se genera al sincronizar la BD con los modelos y generando un *super-usuario* para el sistema. Esta interfaz actúa de forma similar a phpMyAdmin

(herramienta de libre acceso escrita en PHP, y pensada para administrar MYSQL por medio de la WEB), que gestiona la BD por medio de un explorador *Web*. La sección de administración de Django ofrece una interfaz *Web* sencilla e intuitiva por medio de la cual se ingresan datos correspondientes a los modelos de la aplicación y que son guardados directamente en la BD, esto es muy útil cuando se tiene la necesidad de ingresar constantemente información al sistema, además ahorra tiempo para el desarrollador ya que no tiene que realizar rutinas de llenado para cada modelo por separado.

2.2 Sistemas Gestores de Bases de Datos

Una Base de Datos (BD) es un conjunto de ficheros estructurados de tal forma que se genere una relación que facilite la recopilación de la información mediante consultas y que a su vez también más de un usuario tenga la posibilidad de acceder a la misma información. Un Sistema Gestor de Base de Datos (SGBD) es un software encargado de administrar toda la información que existe en una BD, realiza tareas como agregar información, obtener información, eliminar información e inclusive crear nuevas BD utilizando lenguajes como los que se describen a continuación [40]:

- Lenguaje de Definición de Datos *(DDL, Data Definition Languaje).* Está enfocado hacia aquellos usuarios encargados del diseño de las bases de datos. Con este lenguaje el desarrollador puede definir las características de cada elemento que conformará la BD, como los tipos de campos que conforman una tabla, límites de caracteres e información sobre configuraciones de la BD en general.
- Lenguaje de Manejo de Datos *(DML, Data Management Languaje).* Es el que realiza consultas a una base de datos y el encargado de obtener la información deseada por el usuario por medio de sus consultas.
- Lenguaje de Control de Datos *(DCL, Data Control Languaje).* Este lenguaje se hace cargo de administrar las funciones de gestión que realiza el SGBD, tales como la ejecución de *commits* después de cada actualización de datos, se encarga de realizar los *rollback* cuando el sistema ha sufrido de algún problema y entre otras funciones relacionadas con la seguridad de la BD.

- El lenguaje más utilizado actualmente es el SQL (*Structured Query Languaje*), que implementa las características de los lenguajes mencionados anteriormente y que se ha convertido en el estándar utilizado por la mayoría de los SGBD Relacionales [41] [42].

Por otra parte existen actualmente diversos SGBD, a continuación se describen los cuatro más utilizados para la generación de una BD.

2.2.1 Oracle

Oracle Database es un SGBD Objeto-Relacional (SGBDOR), que es un derivado de las BD relacionales pero con la implementación de la programación orientada a objetos, que permite el almacenamiento más complejo en la BD. *Oracle Database* esta creado para ser utilizado en computación GRID (una colección de distintos recursos computacionales distribuidos en diferentes ubicaciones conectados entre sí), la cual es una tecnología que se encarga de crear una relación coordinada entre diversos tipos de recursos. Se considera como una forma de computación distribuida, utilizando varios recursos como diversas arquitecturas de computadoras, sistemas operativos, sistemas de *clústers*, entre otros recursos. La BD se conforma de una estructura física y una estructura lógica, cada una separada de la otra, permitiendo con esto el acceso hacia la estructura física para cualquier mejora de software o almacenamiento sin afectar la estructura lógica, en donde:

- Estructura Lógica. Contiene todos los esquemas, bloques de datos, segmentos y tablas. Todas aquellas estructuras hacen referencia a la forma en cómo se encuentra seccionada la BD y la manera en la cual se accede a ella.
- Estructura Física. Contiene todos los archivos de datos, *logs* y controles de archivos, respaldos y archivos de parámetros. Además la estructura física está conformada por:
    - Archivos de datos (*datafiles*). Todas las *Oracle Databases* contienen uno o más *datafiles*, los cuales contienen toda la información de la BD. Uno o más *datafiles* conforman una unidad lógica de almacenamiento llamada *tablespace*.

- *Tablespace*. Es una unidad lógica de almacenamiento en las cuales se alojan datos del esquema de BD como tablas, índices, entre otros aspectos.
- Archivo de control. Contiene información respecto a la estructura física de la base de datos, como nombre de la base de datos, nombres y locaciones de las bitácoras (*logs*) y archivos de datos (*datafiles*).
- Archivos de parámetros (*Parameter Files*). Contiene una lista de parámetros de configuración para las instancias de la BD y la BD misma. Estos archivos permiten indicar los parámetros de inicialización, como configuraciones de acceso, memoria cache, configuración de índices, etc.
- Archivos de respaldo. Estos archivos son agendados en los *parameter files* de manera que se generen automáticamente los archivos de respaldo de los *datafiles* que constituyen la BD.

Oracle cuenta con diferentes tipos de licencias, desde licencias *Enterprise* para grandes empresas, licencias *Express* que son gratuitas y una versión *Lite* especial para dispositivos móviles [43] [44].

2.2.2 Microsoft SQL server

SQL Server es un sistema gestor de BD relacionales (SGBDR) creado utilizando el lenguaje de programación *Transact-SQL*. *SQL Server* ofrece un mejor manejo de los datos, previniendo la pérdida de los mismos inclusive cuando ocurren errores de sistema. Cada proceso de transición se mueve por diversos controles que guardan las etapas en las cuales se va agilizando el manejo de los datos, estos controles guardan la información en la cual se quedó un proceso y si el sistema llega a tener un problema de ejecución o externo, este control se encarga de restablecer el proceso desde donde se quedó sin perder los datos, además algunas de las bondades ofrecidas por *SQL Server* son:

- *In-Memory OLTP*. Esta función permite optimizar el proceso de las transacciones en línea que se realizan en la BD, esto creando una tabla de acceso extra de memoria optimizada. Las tablas de memoria optimizada son consultadas utilizando *Transact-SQL* de la misma manera que una tabla normal

alojada en memoria, pero con la diferencia que estas tablas son compiladas directamente en lenguaje máquina, aumentando el acceso a la información. Las tablas de memoria optimizada fueron creadas para sistemas con una concurrencia de conexiones muy altas y que requieren de un rápido acceso y respuesta de peticiones.

- *In-memory Column Store Index*. Estos son los índices estándares que utilizan las tablas de *SQL Server* para el acceso a grandes cantidades de información, al igual que MyISAM para MySQL. Aparte del indexado también ofrece una compresión mayor a la de otros motores de búsqueda.
- *AlwaysOn.* La alta disponibilidad de los datos se ha vuelto muy importante para las BD, por lo cual *SQL Server* implementa esta función que ofrece una solución empresarial a los problemas de recuperación de datos y alta disponibilidad de la información. *Always On Availability Groups* funciona sobre un grupo de usuarios de la BD los cuales tienen los permisos de acceso a un ambiente de tolerancia de datos, similar al *safe mode* en el sistema operativo *Windows*, y que da acceso a varias herramientas de recuperación y otras BD utilizadas como respaldos, pudiendo configurar cualquiera de ellas como principal cuando la BD principal presenta fallos.
- *Transparent data encryption (TDE).* TDE es una encriptación a nivel archivo utilizada también por *Oracle*, que se utiliza para contrarrestar la protección de datos en reposo, tales como la encriptación de datos utilizados para los pagos de tarjetas de crédito, donde la información debe ser almacenada temporalmente en la BD para poder realizar las debidas transacciones.

Una de las características únicas de *SQL Server* es la posibilidad de realizar respaldos en la nube utilizando *Microsoft Azure*, la cual a su vez permite la administración de la información que conforma la base de datos. *SQL Server* solo funciona bajo sistemas operativos *Windows* y las funciones antes mencionadas solo se encuentran disponibles en las versiones profesionales y empresariales del software, los cuales requieren de una licencia de paga [45].

2.2.3 PostgreSQL

PostgreSQL es un sistema gestor de BD relacional con licencia BSD (*Berkeley Software Distribution*), la cual es una de las licencias con menores restricciones, dando la oportunidad de utilizar el código fuente en software no libre. A diferencia de MySQL, PostgreSQL utiliza multiprocesos en vez de multi-hilos, en este caso un multiproceso con errores no afecta el rendimiento del sistema. PostgreSQL está optimizado para trabajar con múltiples conexiones y una gran cantidad de datos, cuenta con uno de los sistemas de verificación de integridad de datos más robusta que otros SGBD. Además PostgreSQL cuenta con algunas características particulares como:

- *Nested Transactions*. Las transacciones anidadas en PostgreSQL no permiten que otras transacciones que se encuentran actualmente en proceso tengan acceso a información utilizada por otros procesos, evitando con esto corromper los datos de la BD.
- *Two-Phase Commit Protocol (2PC)*. Este protocolo ofrece un alto grado de confiabilidad debido a que permite deshacer los cambios que se ejecutan al realizar un *commit* si se generara un error de sistema. Un *commit* hacia la BD se encuentra monitoreado por procesos de comunicación entre el cliente y el servidor, donde el cliente al realizar un proceso de guardado envía la información debida al servidor y este al momento de finalizar la actualización y/o guardado debe mandar una señal de término satisfactoria hacia el servidor, indicando que el proceso se ha realizado satisfactoriamente.
- *Write-Ahead Log (WAL)*. Es un esquema de recuperación de sistemas en donde se registran todas las modificaciones permanentes que se realizan en la BD. En estas bitácoras se guardan los archivos que fueron actualizados y se guarda un respaldo de la información anterior, permitiendo con esto realizar una recuperación de datos. Este esquema es también utilizado para guardar los registros generados por las transacciones y poder realizar una recuperación de los datos si se llegaran a necesitar [46].

2.2.4 MySQL

Es un SGBDR que permite almacenar información en diversas tablas y crear una relación entre la información de una con otra por medio de un identificador específico. Esta característica agiliza el acceso de la información en base a la estructura generada por la relación de las tablas de BD. Algunas de sus principales características son [47]:

- El sistema multiusuario de MySQL permite crear usuarios específicos solo para cierta información, evitando el acceso a otros esquemas, esto con motivo de tener una mejor gestión sobre los accesos hacia la BD.
- MySQL implementa una tecnología llamada *Open Data Base Connectivity*, un estándar de acceso a las BD desde un cliente remoto sin importar que SGBD este administrándola.
- Una de las características más destacables de MySQL es la capacidad de poder escoger el motor de almacenamiento de cada tabla por separado. Cada motor de búsquedas tiene diferentes bondades que permiten optimizar el acceso a cada tabla por separado dependiendo de los objetivos para los cuales se hayan creado las mismas.

Dos de los motores de almacenamiento más utilizados en MySQL son MyISAM e InnoDB en donde:

1. MyISAM [48]. Es el mecanismo de almacenamiento que utiliza MySQL por *default* para cada tabla generada siempre que no se haya especificado un motor diferente. Utiliza el mecanismo ISAM (*Indexed Sequential Access Method*), creado por IBM (*International Business Machines Corp.*), que prioriza el acceso a la información por medio de índices, llaves primarias o un índice compuesto por más de un campo. MyISAM ha implementado optimizaciones y tecnologías de búsqueda de información (índices), como *Full-Text Index*, que agilizan más los accesos hacia las tablas que utilizan este motor de almacenamiento basado en texto o cadenas de texto. Un índice o llave hace referencia al campo de una tabla que podrían repetirse y que se configura para agilizar la búsqueda de datos en la tabla por medio de este campo. Una tabla siempre debe tener un campo que actúe como llave primaria, en este caso la llave primaria no debe repetirse y está compuesta por uno o más campos de la tabla. Por otra parte el índice *Full-Text*

Index, se crea sobre uno o más campos de tipo *Full-Text* y agiliza la búsqueda de cadenas de texto. Para realizar una búsqueda de *Full-Text* se utilizan las palabras reservadas de *MySQL MATCH y AGAINST*, donde *MATCH* lleva como argumentos el o los campos que conforman el *Full-Text Index* y *AGAINST* contiene como argumento el texto o cadena que será buscada en los campos de la tabla.

2. InnoDB. Este método de almacenamiento poco a poco va alcanzando la velocidad que tiene MyISAM debido a las grandes optimizaciones que se han implementado. Sin embargo cuando se trata del proceso de una gran cantidad de datos, InnoDB aún sigue teniendo caídas de rendimiento. InnoDB se rige por el modelo ACID, acrónimo de *Atomicity* (Atomicidad), *Consistency* (Consistencia), *Isolation* (Aislamiento), *Durability* (Durabilidad). Estas propiedades juntas forman una serie de procesos llamadas transacciones, que conservan la información almacenada de manera que no se corrompa de ninguna manera. Si al momento de realizar una transacción esta genera un error o el sistema sufre un fallo debido a problemas de configuración o problemas externos, todos los cambios realizados o que se hayan podido realizar se regresan a su estado original. Las propiedades de ACID se describen a continuación:
    a) **Atomicidad**. Es una propiedad que asegura la no corrupción de los datos, por lo que al ocurrir un error al momento de la transacción se asegura que esta no quede incompleta. Una transacción está formada por un conjunto de instrucciones y solo se finaliza de manera correcta si todas las operaciones han sido ejecutadas satisfactoriamente, este principio garantiza que los datos almacenados son correctos.
    b) **Consistencia**. Se encarga de revisar que los datos obtenidos por cada proceso de la transacción sean correctos y sin errores, y al mismo tiempo de revisar que la transacción haya finalizado y obtenido datos legibles y exactos.
    c) **Aislamiento**. Cada proceso de una transacción tiene acceso a información de la BD y durante el transcurso de este proceso esa información no debe ser manipulada ni leída por ningún otro proceso, ya que esto podría corromper la información obtenida por los resultados de

cada proceso y como consecuencia la transacción podría fallar o bien generar datos incorrectos. Por otra parte MySQL tiene niveles de aislamiento que se utilizan para las transacciones a partir de los requerimientos del sistema tales como:

- *Serializable*. Este nivel de aislamiento es el más conservador y tiene la característica de bloquear el recurso utilizado por un proceso de transacción y no permitir el uso del mismo hasta que la transacción haya finalizado por completo.
- *Repeatable Read*. Este nivel es el que utiliza InnoDB de manera automática para sus transacciones y tienen un nivel de aislamiento menor. Este aislamiento no bloquea los recursos utilizados hasta el término de la transacción, si no que los recursos utilizados se van liberando conforme cada proceso de la transacción los termina de utilizar. Este nivel de bloqueo es el más rápido y utilizado para realizar transacciones que no requieren de una mayor prioridad, pero aún la información tiende a ser modificada por otros procesos.
- *Read Commited*. Este nivel de bloque se distingue por ser de los que permiten mayor fluidez de los datos. Todos los recursos utilizados por un proceso son vistos por otro al momento que la información se haya guardado, esto al realizar un *commit* de la información, y no necesariamente cuando el proceso esté finalizado.
- *Read Uncommited*. Este nivel de bloque es el más invasivo, debido a que todos los procesos visualizan y modifican los recursos sin importar que otro ya esté trabajando con ellos. Este nivel casi no se utiliza por la alta tasa de corrupción que llegan a tener los datos de la BD.

d) **Durabilidad o Persistencia.** Esta característica se asegura que una vez almacenados los datos en la BD estos se mantengan correctos y no sufran ningún tipo de deterioro o eliminación, por ejemplo si ocurre un error de sistema. Una vez guardada la información en la BD esta debe permanecer correcta y legible hasta que otro proceso requiera de ella[49].

Por tanto para la realización del sistema MiPYME se utilizó el SGBD MySQL y el lenguaje SQL soportado por la herramienta. Todas las tablas utilizan el motor de búsquedas MyISAM puesto que la información almacenada en la BD no será accesada durante el proceso de llenado de la misma, sino hasta que todas las encuestas estén ingresadas por completo en el sistema, suponiendo MyISAM como motor de búsqueda ideal.

2.3 Servidores Web

Un Servidor *Web* es un software encargado de recibir, responder y denegar las peticiones hechas por los usuarios hacia una página *Web* alojada en el servidor *Web*, de acuerdo a las políticas, reglas y lineamientos que se hayan configurado en los archivos de configuración del servidor [50]. Algunas de las funciones que debe cumplir un servidor *Web* son las siguientes:

- El servidor *Web* gestiona de manera eficiente las peticiones de clientes, ya que en sistemas como *Google* hay una concurrencia de peticiones HTTP/HTTPS simultaneas muy grande, las cuales podrían causar que el servidor se bloquee si no se manejan de forma adecuada. Algunas peticiones también conllevan peticiones a BD, las cuales a su vez podrían causar un colapso en el SGBD.
- Restricciones de acceso a archivos o ubicaciones a las cuales solo administradores tienen acceso o archivos que guardar información privada sobre los diferentes usuarios que estén utilizando la página *Web*.
- Gestionar los errores que se susciten mientras el sistema esté ejecutándose, así como redireccionar al usuario hacia una página de error que indique el estado del sistema y la opción de volver a realizar la petición en determinado tiempo sin necesidad de ingresar de nuevo al sistema.
- Almacenar bitácoras de accesos, errores y toda aquella información que sea necesaria para monitorear el funcionamiento del sistema.

- Al responder a la petición de un cliente se debe dirigir la información obtenida hacia el navegador *Web,* y que este lapueda interpretar de forma correcta y entendible.

Figura 2.6 Diagrama de funcionamiento de un servidor *Web*. FUENTE: [50]

Servidor *Web* se le denomina también al hardware en el cual se ejecuta el software. En la Figura 2.6 se presenta una secuencia del funcionamiento de un servidor *Web*, donde:

1. Desde el cliente de navegación *Web* se ejecutan peticiones hacia el servidor *Web* mediante una conexión a Internet.

2. El servidor *Web* se encarga de procesar una o varias peticiones *Web* provenientes desde diferentes navegadores *Web*, donde el servidor *Web* en base a las configuraciones, módulos y limitantes se encarga de procesar la petición.

3. Una vez procesada la petición se obtiene la información requerida, la cual es accesa por medio de un SGBD. Las BD están regularmente en el mismo servidor *Web*.

4. Una vez procesada la información, el servidor *Web* la prepara de modo que pueda ser leída por los navegadores desde los cuales se hizo la petición.

5. Una vez preparada la respuesta de la petición, se manda de regreso por el puerto de escucha directamente hacia la conexión que realizó la petición.

6. Los navegadores *Web* muestran la información de respuesta del servidor legible y clara, siempre y cuando esta venga bien estructurada y estandarizada.

Actualmente existen diversos servidores Web tales como Apache, NGINX, IIS, Tomcat, Lighttpd, entre otros. A continuación se describen los cuatro servidores *Web* más utilizados.

2.3.1 Microsoft IIS

Es un servidor *Web* de licencia privativa por parte de Microsoft, IIS (*Internet Information Server*), integra tecnologías como ASP.NET, *Windows Process Activation Service* (WAS), un motor de servidor *Web* con la capacidad de agregar o remover módulos según se requieran. Los módulos son características individuales que los servidores utilizan para el procesamiento de solicitudes, como la autenticación de credenciales de clientes o módulos que gestionan la actividad de la memoria caché. Por otro lado IIS agrega un sistema de gestión de solicitudes según del tipo que se trate, esto permite un mejor manejo del servidor ya que cada petición realizada es gestionada por un módulo específico.

Las aplicaciones de IIS se manejan de modo aislado, esto mediante el sistema que limita el número de aplicaciones que se ejecutan al mismo tiempo, evitando con esto que una aplicación sea corrompida por el proceso de otra. El servidor solo corre un modo de aislamiento a la vez, pero con la posibilidad de correr diversos módulos que gestionen sus propias aplicaciones, evitando aún más la posibilidad de que las aplicaciones sean corrompidas por otros procesos.

El módulo integrado de conjunto de aplicaciones (*Integrated Application Pool Mode*) funciona utilizando la recientemente agregada arquitectura integrada de procesos de solicitudes, en la cual una aplicación recibe una petición y a su vez esta es pasada por un conjunto de eventos que verifican cada petición por secciones y al final generan una respuesta. Este proceso es más ágil al modo clásico, eliminando eventos duplicados como verificaciones del cliente y credenciales.

El módulo clásico de conjunto de aplicaciones (*Classic Application Pool Mode*) es el modo más robusto pero más tardado, debido a las verificaciones de seguridad que se repiten, como sistemas de autenticación y verificación, que son ejecutados cada vez que un proceso llega a un evento [51].

2.2.2 Nginx

Es un software encargado de gestionar las peticiones hechas con los protocolos HTTP, HTTPS (Hypertext Transfer Protocol Secure, Protocolo Seguro de Transferencia de Hipertexto), SMTP (Simple Mail Transfer Protocol, Protocolo para Transferencia Simple de Correo), IMAP (Internet Message Access Protocol, Protocolo de Acceso a Mensajes de Internet), entre otros. También utilizado para el balanceo de carga de los servidores y memoria cache. Nginx no crea hilos individuales como lo hace Apache, si no que utiliza un proceso basado en eventos. Nginx divide su trabajo en dos procesos:

1. *Worker Connections* (Conexiones de Trabajo). Se encarga de manejar las peticiones hechas por los usuarios y las respuestas obtenidas de esas peticiones. Una sola conexión de trabajo gestiona alrededor de 1,024 conexiones al mismo tiempo.

2. *Worker Process* (Procesos de Trabajo). La cantidad de procesos de trabajo varían conforme a las cualidades del servidor y los requerimientos del sistema. A demás, cada proceso de trabajo gestiona diferentes tipos de peticiones, de esta manera se tiene un sistema más entendible y depurado. Los procesos de trabajo envían las solicitudes al *Nginx Master Process* que a su vez envía la respuesta al cliente.

Nginx trabaja de manera asíncrona, por lo que ningún proceso es afectado por otro, dándole más fluidez al sistema y utilizando memoria virtual compartida hace que Nginx utilice notablemente menos memoria del servidor que otros servidores. Además de ser un servidor web de libre acceso y multiplataforma que optimiza el uso de los núcleos e hilos de procesamiento del servidor de manera que todos los procesos y peticiones son ejecutados con mayor rapidez [52].

2.2.3 LiteSpeed

Es un servidor *Web* de alto desempeño con una licencia estándar y una empresarial (ambas licencias con un costo), compatible con muchas de las características de Apache, incluidos algunos módulos de reescritura de direcciones

(*mod_rewrite*), archivo de configuración de acceso *(.htaccess*), y el módulo de seguridad (*mod_security*). LSWS (LiteSpeed Web Server) carga directamente los archivos de configuración de Apache, así como a Apache completamente, debido a su alta compatibilidad. LSWS implementa una arquitectura orientada a eventos que agilizan el rendimiento del sistema sin la necesidad de utilizar mucha memoria tanto RAM como del CPU. Este servidor *Web* toma ventaja de su código optimizado para uso de contenido estático y que realiza peticiones al núcleo del servidor (*kernel*) de manera más rápida. En cuanto a seguridad, LSWS bloquea los ataques DoS (*Denial of Service*) y DDoS (*Distributed Denial of Service*) sin la necesidad de hardware o configuración externa. Y aprovechando la compatibilidad es capaz de utilizar las mismas configuraciones de seguridad ya hechas para HTTPD e inclusive la misma sintaxis. LSWS tiene herramientas pre-instaladas que automatizan las páginas *Web* que están escritas bajo PHP. Otras de las herramientas que implementa LSWS se mencionan a continuación:

- *Hosting* Virtual. Soporta múltiples *hosts* referenciados por *ip* o nombre, y con herramientas como *cPanel* y *Plesk* se obtienen opciones para controlar los *hosts*.
- *IP Level Throttling*. Esta herramienta modula el tráfico del ancho de banda que tiene el servidor y controla de igual forma el tráfico que sale hacia las IP estáticas. Esta herramienta impide que el servidor sea objetivo de ataques DoS y DDoS sin la necesidad de utilizar una herramienta externa.
- *Request Filtering*. Es otra herramienta de seguridad compatible con el módulo de seguridad de Apache (*mod_security*) y que implementa una protección en contra de ataques *SQL/Script Injection* (Inyección SQL) y XXS (*Cross-Site Scripting*), los cuales son ataques que se llevan a cabo por medio de formularios HTML generados en el cliente *Web* (explorador *Web*) y que tienen la finalidad de obtener datos personales, tales como contraseñas, direcciones, claves de tarjetas de crédito, entre otros.

LSWS es un servidor *Web* de alta disponibilidad que se recupera de manera instantánea de errores de sistema. Además LSWS se ejecuta completamente en el espacio de memoria de un usuario, no en el administrador, dando al usuario la oportunidad de manejar diversas versiones del servidor sin afectar la funcionalidad del sistema operativo [53].

2.2.4 Apache

Es un servidor con licencia abierta, tanto del uso del código como de la modificación del mismo. Apache HTTP Server (*HTTPD*) es un proyecto de *The Apache Software Foundation* y fue dado a conocer en 1995 [1], convirtiéndose en el servidor *Web* más utilizado desde abril de 1996 a la fecha [54]. HTTPD es un sistema multiplataforma que viene pre-instalado en sistemas basados en Unix, e instalable para sistemas Windows y Mac OS. HTTPD implementa los protocolos de solicitudes (*requests*) más recientes y más utilizados por los desarrolladores *Web*, con esto Apache se mantiene a la vanguardia en cuanto a tecnologías *Web*. Los módulos de Apache se encargan de procesar de manera separada los diferentes tipos de solicitudes que se realizan hacia el servidor. La API de módulos de Apache es utilizada para crear módulos no oficiales, hechos por terceros (*third-party*), y activarlos para trabajar en conjunto con HTTPD, e inclusive se modifican los módulos oficiales para satisfacer las necesidades de los desarrolladores, esto gracias a su licencia sin restricciones. Debido a su licencia abierta, Apache tiene una gran cantidad de desarrolladores que trabajan en conjunto para la reparación de *bugs,* problemas de seguridad y problemas en el funcionamiento del sistema, haciendo de Apache un sistema muy robusto y de constante mejora y actualización. Algunas de las peticiones que soporta HTTPD son las siguientes [55]:

- Implementación de autenticación por BD, BD Relacionales, e inclusive utilizando el protocolo ligero de acceso a directorios (LDAP), que funciona de manera similar a una BD aunque utiliza un sistema de almacenamiento diferente.
- Tiene la opción de proteger con clave las páginas que se indiquen en la configuración para que solo ciertos usuarios tengan acceso a ellas, como administradores o desarrolladores. Todo esto directamente en HTTPD y sin necesidad de desactivarlo.
- HTTPD tiene uno de los mejores manejadores de errores, dándole al administrador del servidor herramientas como re-direccionamiento de páginas en caso de algún error, mostrar páginas personalizadas de error, entre otros.

- HTTPD ofrece la opción de indicar diferentes directorios de sistema, esto con el fin de tener una mejor administración de la página, separando por ejemplo los directorios de documentos, media y código en diferentes ubicaciones.

- HTTPD está pensado para soportar más de una configuración de página *Web*. Utilizando *Virtual Hosts* (Anfitriones Virtuales, *vhosts*), tal y como se observa en la Figura 2.7. HTTPD soporta la configuración de diversas páginas *Web* utilizando el mismo servidor.

- HTTPD lleva un sencillo y configurable sistema de bitácoras de sistema (*logs*), aplicados a errores de servidor, o los accesos al servidor que son fácilmente depurados para monitorear el funcionamiento del servidor. Apache permite la configuración de bitácoras para *vhosts* de manera separada, para monitorear cada página *Web* en sus propias bitácoras.

Figura 2.7 Virtual Hosts de Apache. FUENTE: [55]

Apache está optimizado para un balance entre flexibilidad, portabilidad y rendimiento, por lo que comparado con otros sistemas Apache podría verse vencido en cuanto a rapidez de respuesta con respecto a Nginx o rapidez de configuración en comparación con IIS, pero todo esto Apache lo retribuye con un sistema siempre actualizado y robusto, que para muchas empresas es el rasgo más importante que debe ofrecer un servidor *Web*. Por lo tanto para desplegar el sistema MiPYME-CAEO se utilizó el servidor Apache debido a sus ventajas y características descritas anteriormente, a la facilidad de configuración y el previo conocimiento obtenido sobre la herramienta.

Capítulo 3. Metodologías Ágiles

Introducción

Actualmente el desarrollo de software no es una actividad sencilla, sin embargo a pesar de ello han surgido metodologías que nos facilitan esta ardua tarea como lo son las metodología tradicionales las cuales se enfocan en el control de procesos, teniendo actividades fijas, resultados a producir y los elementos o herramientas a utilizar. Estas metodologías han presentado algunas dificultades al implementarlas, tales como una alta tasa de errores y periodos de desarrollo muy grandes.

	Metodologías Ágiles	Metodologías Tradicionales
Descripción	Surge buscando simplificar el desarrollo de software, así como reducir los costos	Se definen tareas y equipos de desarrollo para cada una de ellas
Metodología de procesos	Planificación, Diseño, Codificación, Pruebas, Especificación de casos de uso	Análisis y diseño, Requerimiento, Prueba Desarrollo
Enfoque	Adaptativo	Predictivo
Tamaño de proyecto	Pequeño	Grande
Duración de proyecto	Corto	Largo
Documentación	Poca	Pesada
Énfasis	Orientada a las personas	Orientada a los procesos
Ciclos	Muchos	Limitados
Planificación	Mínima	Exhaustiva
Retorno de Inversión	A principio del proyecto	Al finalizar el proyecto
Equipos de trabajo	Pequeños	Grandes
Carga de trabajo	Poca	Moderada a Grande
Detección de errores	Antes y después de cada iteración	Preferentemente durante su desarrollo, errores detectados en etapas avanzadas pueden poner en riesgo el proyecto completo
Soporte técnico	Constante	Poco

Figura 3.1. Comparación entre las metodologías tradicionales y ágiles. FUENTE: [57]

Con la intención de poner mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones cortas, surgen las metodologías ágiles, las cuales han demostrado su efectividad en relación a proyectos con requisitos cambiantes y cuando es necesario ajustar los tiempos de desarrollo, pero manteniendo una alta calidad, por lo cual estas metodologías ágiles han revolucionado favorablemente el desarrollo de software. En la Figura 3.1 se muestra una comparación entre las tecnologías tradicionales y las metodologías ágiles, en donde se observan las diferencias que tienen las metodologías ágiles con respecto a las metodologías tradicionales.

Las metodologías ágiles valoran principalmente al cliente y la interacción que tengan con el equipo de desarrollo, el desarrollo de software funcional y robusto, la colaboración con el cliente y responder eficientemente a los cambios y/o modificaciones que realicen a los objetivos del sistema. Con la implementación de tecnologías ágiles en el desarrollo de software la tasa de errores en los sistemas descendió [56], lo que comprueba que las tecnologías ágiles son la mejor opción en cuanto a desarrollo de software se trata. Se ha propiciado el surgimiento de diversas metodologías ágiles, entre las que se encuentran SCRUM, *Crystal Methodologies, Dynamic System Development Method (DSDM), Feature Driven Development (FDD), Lean Development (LD), eXtreme Programmming (XP)* y *Adaptative Software Development (ASD)*. A continuación se describen las cuatro metodologías ágiles más utilizadas en los últimos años para el desarrollo de software.

3.1SCRUM

SCRUM es una metodología para gestión de proyectos en la cual influyen factores importantes como agilidad, incertidumbre y flexibilidad. Todas estas son características primordiales que debe cumplir una metodología ágil. Surge después de la investigación de Takeuchi y Nonakaen, quienes se dieron cuenta que algunas empresas tecnológicas que se encontraban en el mismo rango que otras tantas, terminaban en mucho menos tiempo sus productos y además sus productos eran de calidad y generaban menos inversión al momento de su manufacturación [58].

SCRUM es una metodología de desarrollo simple, pero que requiere de mucho

esfuerzo y trabajo debido a que no se sigue un plan detallado, en vez de esto se debe adaptar continuamente conforme evoluciona el proyecto en cada una de sus iteraciones. Lo cual indica que es una metodología completamente orientada a las personas. Cada iteración debe ser diaria y deberá ser puesta a revisiones, en las cuales se determinará el avance del proyecto y el siguiente paso a seguir. El proceso de SCRUM inicia cuando se define a grandes rasgos el proyecto y la funcionalidad de todas aquellas características principales que debe cumplir, estas deben poder llevarse a cabo en lapsos llamados periodos o iteraciones, tal y como se muestra en la Figura 3.2, cada periodo o iteración debe finalizar con un avance operativo del proyecto, es decir que se debe terminar cada periodo con una parte funcional del proyecto, la cual haya sido probada y que además tenga la posibilidad de ser modificada según se requiera en el futuro. Estas iteraciones deben ser revisadas diariamente en reuniones donde todos los equipos revisen los avances del proyecto y se determine el trabajo a realizar para la siguiente reunión.

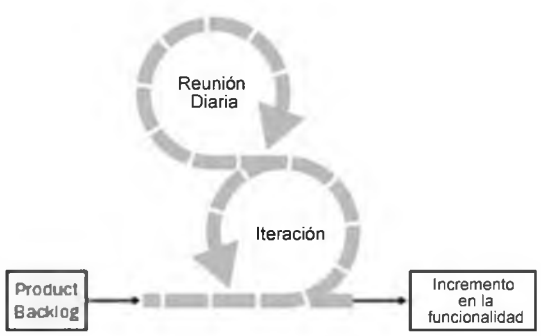


Figura 3.2 Esquema de funcionalidad SCRUM. FUENTE: [58]

El control de la evolución del proyecto debe ser adaptable y no debe atentar contra la autonomía de los equipos, dejando a cada uno la posibilidad de proponer soluciones determinadas en base al objetivo que se busca cumplir en cada iteración. Para esto se deben cumplir con las siguientes características:

1. Revisión de iteraciones. Al final de cada iteración se realiza una reunión con todo el personal del proyecto. Cada iteración debe tener una duración máxima de 30 días.
2. Desarrollo incremental. Implica que al final de cada iteración se obtenga un avance funcional del sistema que pueda ser evaluado.
3. Desarrollo evolutivo. La predicción en un principio del producto final y las partes que lo formarán no se pueden aplicar a proyectos donde la incertidumbre

es una de las principales variables. SCRUM se aplica a proyectos en los cuales solo se tiene un objetivo que no se basa en características específicas, en vez de esto el proyecto esta propenso a cambios debido a la inestabilidad de los requisitos. SCRUM diseña y genera la estructura del proyecto conforme él mismo va evolucionando.

4. Auto-organización. Los factores impredecibles que surgen durante el desarrollo del proyecto son analizados por cada grupo y tienen la autoridad suficiente para determinar soluciones oportunas que favorezcan la resolución de las problemáticas.
5. Colaboración. Cada persona involucrada en el proyecto tiene la posibilidad de proponer la solución a una problemática sin importar que su equipo no se esté encargando de ella. Cada miembro de los equipos debe colaborar con los demás según sus capacidades y no en base al puesto y equipo en que se encuentre [59].

Por otra parte, los elementos que conforman SCRUM son los siguientes:

1. *Product Backlog*. Esta es la lista priorizada de las características que debe cumplir el proyecto. Esta lista será gestionada por el cliente y el administrador SCRUM de la empresa, quien le indicará los costos y tiempos que se requieren para cumplir un objetivo del proyecto. Además contendrá todas aquellas características y observaciones que ayuden a realizar el proyecto.
2. *Spring Backlog*. Es la lista de objetivos que deberán realizarse para completar cada iteración. Estos objetivos serán divididos en sub-tareas más pequeñas que serán asignadas a cada persona o personas pertinentes para su correcto desarrollo.
3. Incremento. Esta lista representa cada objetivo cumplido de las iteraciones y que a su vez se considera funcional. Estos objetivos cumplidos son referidos al cliente y a su vez el cliente decide si el objetivo cumple con sus necesidades o considera si requiere ajustes.

SCRUM está pensado para ser aplicado en aquellas empresas en las cuales no se tiene un plan de trabajo específico que se deba seguir para el desarrollo de un proyecto, donde los grupos de trabajo tienen una mayor autonomía sobre sus actividades y a su vez donde cada persona involucrada en el grupo de trabajo tiene la posibilidad de

transferir sus conocimientos a los otros integrantes del grupo.

3.2 Dynamic System Development Method (DSDM)

DSDM (Método de Desarrollo Dinámico de Sistemas) es un *Framework* para el desarrollo ágil de software basado en nueve principios esenciales que deben ser seguidos al pie de la letra, ya que pasar por alto cualquiera de ellos puede poner en riesgo la integridad del proyecto. Sin embargo, dependiendo de las necesidades y estructura del proyecto se podrían pasar por alto o modificarlos según se requiera. Los nueve principios se mencionan a continuación:

1. La interacción del usuario en el desarrollo del sistema es primordial. Este principio es el más importante, ya que reduce los errores generados por el usuario y con esto los costos necesarios para lidiar con los mismos. Es recomendado trabajar con un grupo pequeño y seleccionado de usuarios pactando constantes reuniones de trabajo en las que revisarán los avances del proyecto. El grupo de usuarios se encargará de verificar la funcionalidad de los avances, retroalimentando a los desarrolladores con información que servirá para desarrollar un sistema más amigable al usuario final.
2. Los grupos de trabajo deben poder tomar decisiones. Al tener los grupos la posibilidad de tomar decisiones se agiliza el proceso de desarrollo, ya que se evita la pérdida de tiempo que conlleva el esperar que los directivos aprueben cada una de las actividades necesarias para la resolución de problemáticas que ocurran durante el proceso de cada avance.
3. Entrega frecuente de avances. La entrega frecuente de avances promueve la fácil detección de errores. Esto aplica al proyecto y a la documentación donde se especifican los requerimientos y modelos de datos.
4. Las aptitudes de los integrantes de los grupos conllevan a la entrega de avances con pocas o nulas observaciones y favorecen la creación de un proyecto que cumple enteramente con las necesidades del cliente. La buena aplicación de la metodología DSDM permite que el proyecto final sea fácil de actualizar y dar soporte en un futuro.
5. Desarrollo incremental e iterativo. Al descomponer el proyecto en varios objetivos se reduce su complejidad, por lo tanto la manejabilidad y facilidad de

soporte al sistema aumenta. Desde un principio se debe tener en mente que todo proyecto está sujeto a cambios, por lo que cuanto más sencillo se mantenga el desarrollo del sistema, más rápido que se podrán realizar los cambios y actualizaciones.

6. Cada cambio realizado durante el proceso de desarrollo debe poder ser revertido a una versión funcional anterior. Siguiendo la idea de que todo proyecto está sujeto a cambios, cada actualización al sistema debe poder ser revertido, esto para no afectar la funcionalidad del sistema si se genera un problema con la nueva actualización o por si es necesario un cambio en los requerimientos.
7. Para limitar la libertad con la cual son modificados los requerimientos principales del sistema, algunos de ellos deben plantearse como alto nivel desde el comienzo del proyecto. Estos requerimientos de alto nivel no son modificados y deben ser respetados durante todo el proceso de desarrollo del proyecto.
8. Muchos métodos de desarrollo de software dejan el periodo de prueba para cuando se tiene un proyecto casi final, esto es, se realizan las pruebas sobre un proyecto que ya tenga interfaz de usuario y se encuentren trabajando en conjunto las secciones principales del proyecto. Para DSDM es requisito el realizar pruebas cada entrega de avances, dejando al usuario trabajar como normalmente lo haría, de esta manera el sistema se mantiene con una tasa de errores muy baja.
9. Es necesario mantener una relación constante para DSDM, esto es un punto crucial para el desarrollo y termino del proyecto. Cuando varios grupos de trabajo no tienen la suficiente cooperación entre ellos es más probable que el sistema tenga errores debido a que no existe una buena comunicación y por lo tanto se genera una desconfianza [60].

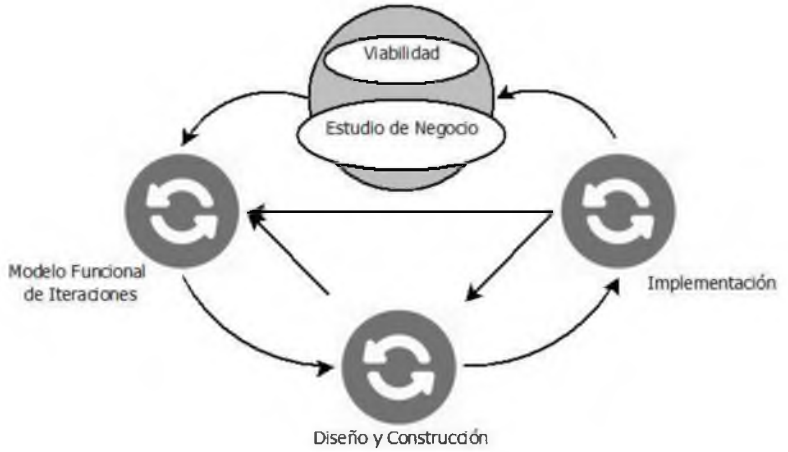


Figura 3.3 Fases de desarrollo DSDM. FUENTE: [60]

Los proyectos realizados con DSDM se dividen en tres fases:

1. Pre-Proyecto. En esta fase es necesario que se dejen en claro los objetivos principales del proyecto. A demás de crear un proceso de planeación preliminar el cual deberá seguirse durante el desarrollo del sistema y que está sujeto a cambios. También durante esta fase se determina si todas las características que solicita el cliente para el proyecto son posibles, dando la oportunidad de modificarlas desde un principio.
2. Adecuación del proyecto. Esta fase comprende los puntos principales que plantea DSDM para el desarrollo del proyecto, como se muestra en la Figura 3.3, se divide en las siguientes sub fases:
    a. Fases Secuenciales. Estudian los sectores referentes al ambiente comercial del negocio y se desarrolla un análisis preliminar del sistema:
        i. Estudio de Viabilidad. Similar al estudio de viabilidad clásico, en el cual se estudia el éxito o fracaso de un sistema en base a determinados parámetros, sin embargo este estudio se basa más en la viabilidad del proyecto a partir del planteamiento DSDM y los planes desarrollados para las siguientes fases.
        ii. Estudio de Negocio. En esta fase se determinan los requerimientos de alto nivel que formarán el proyecto, se desarrolla la arquitectura del sistema y se crea un plan de desarrollo.
    b. Fases de Iteración (El Ciclo de Desarrollo). Fase de desarrollo incremental, diseño y codificación, con un constante despliegue del

sistema utilizando un prototipo evolutivo. Se divide en las siguientes sub-fases:

i. Modelo funcional de iteraciones. Se centra en requerimientos seleccionados específicamente por su prioridad. Cada iteración debe concluir con la terminación de los requerimientos principales del proyecto.
ii. Iteración de Diseño y Construcción. Cada iteración debe terminar con un prototipo entregable de incrementos del sistema.
iii. Implementación. Se centra en ir mezclando todas las iteraciones con el fin de obtener un producto que pueda ser utilizable y validado por los clientes.

3. Post-Proyecto. Una vez que el proyecto está terminado se sigue un proceso donde se verifica el correcto funcionamiento del sistema y si se requiere, realizar modificaciones necesarias. Así como agregar o modificar requerimientos del sistema si el cliente lo solicita [61].

3.3 eXtreme Programming (XP)

XP surge como una nueva metodología de desarrollo de software ágil y simple. Su propósito principal es terminar a tiempo los sistemas de los clientes sin importar los cambios tardíos que se presenten. La interacción tanto del cliente como los desarrolladores del sistema es primordial, ya que el trabajo en equipo y constante comunicación con los clientes que utilizarán el sistema permiten la creación de un sistema de alta calidad.

Todo proyecto realizado con la metodología XP debe definir cuatro variables (costo, tiempo, calidad y alcance), de las cuales tres de ellas podrán ser definidas por los clientes. La cuarta deberá ser definida por el grupo de trabajo (desarrolladores) de la empresa.

1. Costo. El precio total del desarrollo del sistema. Esto debe abarcar todos los gastos que se necesiten para el desarrollo del sistema, tales como las reuniones realizadas durante cada iteración, pago de los desarrolladores, entre otros.
2. Tiempo. El tiempo estimado que será necesario para terminar el proyecto. Se

deben tomar en cuenta todos los objetivos y los posibles cambios que el cliente pueda llegar a solicitar.

3. Calidad. Indica que no necesariamente muchas personas deben estar involucradas en el equipo para desarrollar el sistema. Por lo que 10 o 15 personas son suficientes para realizar un sistema de alta calidad. La calidad del sistema será definido por los estándares y buenas prácticas utilizados por los programadores. Las constantes pruebas del sistema permiten tener un sistema robusto y fácil de modificar o actualizar.
4. Alcance. Se define al estimar las tareas necesarias para satisfacer las necesidades del cliente y decidir de entre ellas las más importantes y que serán desarrolladas primero para el sistema [62].

El Modelo XP, tal y como se presenta en la Figura 3.4, propone un método de trabajo basado en iteraciones. Debido a que muchas veces los clientes no saben definir sus necesidades y esto podría retrasar el proceso de desarrollo, cada iteración debe realizarse en un periodo de tiempo corto. Por lo general cada proyecto debe tener de 10 a 15 iteraciones [63].

Figura 3.4 Diagrama de funcionalidad, Modelo XP. FUENTE: [63]

El desarrollo de un proyecto se separa en cuatro fases:

1. Exploración. En esta fase el cliente define las características y objetivos que desean para su proyecto. Los programadores deben estimar los tiempos en base a estos objetivos y su funcionalidad y dar una fecha estimada de terminación y entrega. La fecha de entrega se estima en base a los objetivos entregados por los clientes, pero son solo estimaciones, propensas a cambios según se estudien más a fondo por los programadores. Al finalizar esta fase se obtiene una visión general del proyecto y los tiempos estimados para su entrega.
2. Planificación. Esta sección está destinada para generar un itinerario de entregas para las iteraciones. Además se definen los objetivos principales y el orden en el

cual deberán ser desarrollados.

3. Iteraciones. Es la fase principal de XP, cada iteración deberá contar con la constante comunicación del cliente y los desarrolladores, ya que de esto dependerá la entrega sin errores de cada iteración. Cada iteración debe tener módulos de pruebas realizadas por los clientes, de esta forma se verifica que el entregable sea funcional y no contenga errores. Cada iteración se encarga de recabar información necesaria para la siguiente iteración. La entrega de iteraciones sin errores indican el correcto funcionamiento del equipo de trabajo y habla de una perfecta comunicación con los clientes.
4. Puesta en Producción. Esta es la fase final, donde el proyecto es desplegado en el servidor donde se encontrará alojado o si se trata de un ejecutable se obtiene como producto final un sistema instalable y listo para ser utilizado. Muchos clientes prefieren ejecutar pruebas finales antes de lanzar a funcionar el sistema, por lo que esta fase podría tener errores de sistema o bien servirá para que el cliente decida si el sistema está listo o desea que se modifiquen características del sistema.

3.4 Adaptative Software Development (ASD)

La metodología que se utilizó para el desarrollo del sistema MiPYME-CAEO fue *Adaptative Software Development* (ASD), debido a sus principales características: iterativa y tolerante a cambios. El proyecto se desarrollo en base a propuestas iniciales y revisiones mediante las cuales se decidieron las optimizaciones pertinentes que fueron surgiendo. Esta metodología destaca por su capacidad de retroalimentar el sistema y la interacción directa con el cliente, creando con esto un sistema que se enfoca de manera más específica a las necesidades del usuario. ASD, a diferencia de otras metodologías como *Crystal Methodologies,* no requiere de un mínimo de 3 desarrolladores; ni de trabajo en equipo como *Feature–Driven Development (FDD),* que fue concebido para trabajos de más de cincuenta desarrolladores. El ciclo de vida que propone la metodología ASD, tal y como se observa en la Figura 3.5, tiene tres fases esenciales:

1. Especulación.- En esta fase se inicia el proyecto y se planifican las características del software.
2. Colaboración.- En esta fase se desarrollan las características.

3. Aprendizaje.- En esta fase se revisa su calidad y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo, además se enfatiza en la velocidad de desarrollo para crear un producto de alta calidad, se realiza un bajo mantenimiento debido a que se involucra al usuario lo más posible, asímismo, usa información disponible de los cambios para mejorar el funcionamiento del software [64].

Figura 3.5 Ciclo de vida ASD. FUENTE: [64]

Las iteraciones se deben plantear desde un principio, pero teniendo en cuenta que solo se hacen en base a especulaciones y que en un futuro podrán cambiar, dando paso a un objetivo mejor estructurado y con bases más fuertes. Cada nueva implementación debe ser llevada a cabo con una gran colaboración entre los miembros del equipo (desarrolladores, usuarios, jefes de área), de esto dependerá que cada objetivo de iteración sea completamente funcional y sin errores. Al final de cada iteración, el grupo de desarrollo se reúne para revisar los objetivos y compartir los métodos y técnicas utilizadas para el desarrollo de los objetivos, esto permite a los demás miembros del grupo aprender de la experiencia de sus demás compañeros. Los procesos que se llevan acabo en la metodología ASD, tal y como se observan en la Figura 3.6, son los siguientes:

1. Inicio de proyecto. En esta fase se definen los objetivos principales del proyecto, se designan los equipos pertinentes para cada objetivo y se definen criterios de desarrollo específicos para mantener un estándar de desarrollo.
2. Fases de desarrollo iterativo. Estas fases comprenden el ciclo de desarrollo del proyecto:
    a. Planeación de ciclos adaptativos. Se determina el tiempo que deberá durar el desarrollo del proyecto, así como el tiempo de duración de cada iteración. Cada iteración debe durar entre dos y ocho semanas. Se

definen los estándares utilizados para el desarrollo de cada objetivo y los componentes que lo conformarán.

b. Desarrollo concurrente de los componentes. Se desarrollan los componentes que conforman un objetivo. Por lo general cada componente es desarrollado paralelamente con los otros componentes del objetivo por los grupos de trabajo designados. En esta fase se deben monitorear y controlar constantemente los cambios y avances, de forma que los objetivos queden listos para la fase de control de calidad.

c. Revisión de calidad. Fase en la cual se realizan pruebas a los objetivos ya funcionales, esto por medio de sesiones realizadas por los clientes. La retroalimentación obtenida por los clientes es inspeccionada por los jefes de cada grupo y se deberán realizar modificaciones y/o correcciones según lo requiera el sistema y los clientes. Al final de cada revisión se debe plantear el siguiente objetivo, así como designar los grupos para cada componente. También se verifica el desempeño de cada equipo y la efectividad de los métodos utilizados. Si surgieron algunos problemas, estos son solucionados de manera que no afecten las siguientes iteraciones.

3. Revisiones finales y liberación del proyecto. Durante este periodo se supone un proyecto ya terminado y se realizan pruebas finales para todos los módulos ya trabajando en conjunto. Se evalúan los resultados y se corrigen los problemas encontrados. Una vez tomada la decisión de liberar el proyecto para su uso, se realizan periodos de capacitación para el personal que utilizará el sistema y se prepara la documentación del proyecto [65].

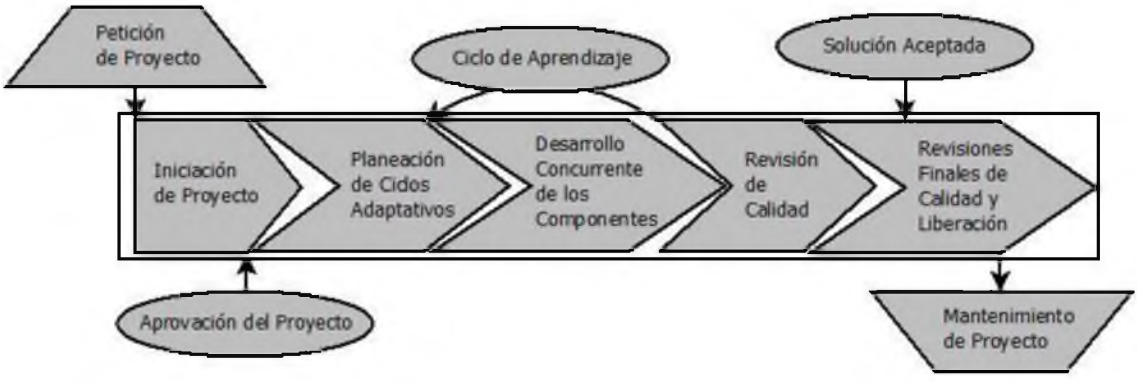


Figura 3.6 Proceso de desarrollo. ASD. FUENTE: [64]

Como todas las metodologías ágiles, la principal ventaja de ASD es su desarrollo basado en iteraciones, lo cual facilita el desarrollo del proyecto, facilita la detección de errores y si se requieren cambios o modificaciones a los componentes se realizan sin comprometer los demás módulos del proyecto. Una característica única que tiene ASD es la posibilidad de compartir la experiencia de un desarrollador o grupo de desarrolladores con los demás, permitiendo que los desarrolladores mantengan un alto nivel de conocimiento y una rápida capacitación. El trabajo en equipo ayuda mucho a terminar los proyectos en un menor tiempo, pero si no existe una buena co-relación entre los integrantes del grupo y/o los clientes, esta característica se vuelve una desventaja para cualquier metodología ágil. Debido a que las metodologías ágiles fueron desarrolladas para dar respuestas rápidas, esto obliga a tener un alto grado de monitoreo durante todo el proceso de desarrollo y constante comunicación. Esto no siempre es posible ya que muchas veces los clientes no tienen el tiempo suficiente para participar en las pruebas que se deben realizar cada iteración.

Capítulo 4. Desarrollo

Introducción

En este capítulo se presenta la aplicación de la metodología ágil ASD que se utilizó para el desarrollo del sistema MiPYME-CAEO, en donde se describen las actividades realizadas en las tres fases que la componen: especulación, colaboración y aprendizaje, indicando desde la generación de la base de datos, reuniones realizadas con el CAEO, período de prueba del sistema, entre otros.

4.1 Fase 1.- Especulación

En esta fase se inició el planteamiento del CAEO relacionado con la insuficiencia de datos disponibles para la detección de MiPYMES en la ciudad de San Juan Bautista, Tuxtepec, Oaxaca, en donde se propuso el desarrollo del sistema "MiPYME-CAEO" que permita la generación de una BD de las MiPYMES asentadas en el primer cuadrante de la ciudad. A continuación se describen las características principales que el CAEO planteó para este sistema.

4.1.1 Encuesta

La parte inicial en esta fase fue familiarizarse con toda la estructura y partes de la encuesta, analizar los diferentes tipos de preguntas, identificar los filtros y las preguntas obligatorias en cada variable propuesta. La encuesta como herramienta principal para la obtención de los datos requeridos debe tener funcionalidades (filtros), que permitan a los encuestadores el llenado rápido de la misma. Entre estas características podemos destacar las siguientes:

- La manera automática en la cual algunas preguntas son omitidas según respuestas obtenidas en preguntas anteriores. Un ejemplo de esto es la pregunta 18, como se muestra en la Figura 4.1, la cual indica que en caso de seleccionar la opción "Ninguna", se deberá pasar directamente a la pregunta 20, omitiendo

completamente la pregunta 19. Al agregar este filtro automático, el llenado de la encuesta se agiliza, ya que los encuestadores no tendrían que preocuparse por aplicar manualmente estos filtros.

18.- ¿Cuál de las siguientes recomendaciones le han hecho sus clientes y de qué forma se la hacen llegar? (Puede señalar más de una opción). En caso de ser "Ninguna" pasar a la pregunta 20.

Mejorar el servicio al cliente		Mejorar la atención al cliente		Mejorar la calidad del producto		Mejorar la presentación del local		Ampliar la gama de productos y/o servicios		Ninguna	Otra:	
Verbal	Escrita	Verbal	Escrita	Verbal	Escrita	Verbal	Escrita	Verbal	Escrita		Verbal	Escrita

Figura 4.1 Estructura de una pregunta con filtro

- Algunas de las preguntas sugieren el poder escoger más de una respuesta, como se muestra en la Figura 4.2, en la pregunta 25 se da más de una opción de respuestas para las secciones de "Si" y "No", para cualquiera de las dos opciones se puede seleccionar más de una respuesta. El filtro en este tipo de preguntas se aplica evitando seleccionar las preguntas de la opción no escogida, por ejemplo, al escoger "Si", todas las respuestas que pertenecen a "No" serán bloqueadas, con el fin de evitar que por error sea una de ellas seleccionada.

25.- ¿Considera adecuada la ubicación de su negocio y a qué factores lo atribuye? (Puede señalar más de una opción)

Si	Local céntrico	Cercanía de otros negocios	Cercanía de hogares	Tránsito constante de personas	Poca competencia	Seguridad	Otros:
No	Lejanía del local	Lejanía de otros negocios	Lejanía de hogares	Poco tránsito de personas	Mucha competencia	No hay seguridad	Otros:

Figura 4.2 Pregunta con opción múltiple de respuestas en base a dos perspectivas diferentes

- Una de las preguntas importantes dentro del cuestionario como se observa en la Figura 4.3, es la número 26 ya que indica si se deberán llenar los anexos en la encuesta. Al seleccionar "Si" en esta pregunta, se debe indicar si el negocio cuenta con área de Mercadotecnia y/o Ventas, las cuales requieren de llenar el anexo A y B, respectivamente.

26.- El negocio cuenta con un departamento de Mercadotecnia y/o Ventas. Si: Mercadotecnia ( ) Ventas ( ) No ( )
En caso de ser "Si" contestar los Anexos A y/o B, respectivamente.

Figura 4.3 Estructura de una pregunta con filtro a los anexos

Estos anexos se deberán llenar al terminar la última pregunta de la encuesta (pregunta 95). Esta es una característica de la cual el encuestador no debe preocuparse debido a que automáticamente el sistema mostrará el o los anexos para

ser llenados. Por otra parte, así como algunas preguntas son omitidas en base a respuestas de preguntas anteriores, también hay preguntas que no deben ir vacías, por lo que el sistema debe ser capaz de detectar aquellas preguntas obligatorias e indicar un mensaje de advertencia para llenar esa pregunta, de otra manera la encuesta no puede continuar.

4.1.2 Graficación

Al poder graficar las respuestas obtenidas para las preguntas, el CAEO obtendrá mejores conclusiones sobre la situación actual de las empresas. Por esta razón las gráficas deben ser perfectamente legibles y mostrar datos precisos. La manera en que se obtendrán las gráficas es la siguiente:

a) Gráficas separadas por pregunta. En este tipo de graficas se deben generar de acuerdo a los siguientes criterios: preguntas solo con una o más respuestas posibles, generarán una sola gráfica. Preguntas con múltiples respuestas en base a una perspectiva distinta generarán 3 gráficas, la primera se dividirá según la perspectiva, como en el caso de la pregunta 25 que se muestra en la Figura 4.2, en este sentido la primera gráfica se representará por "Si" y "No", la segunda gráfica será representada por las respuestas de la perspectiva "Si" y la tercera gráfica por las respuestas que representen la perspectiva de "No", tal y como se muestra en la Figura 4.4.

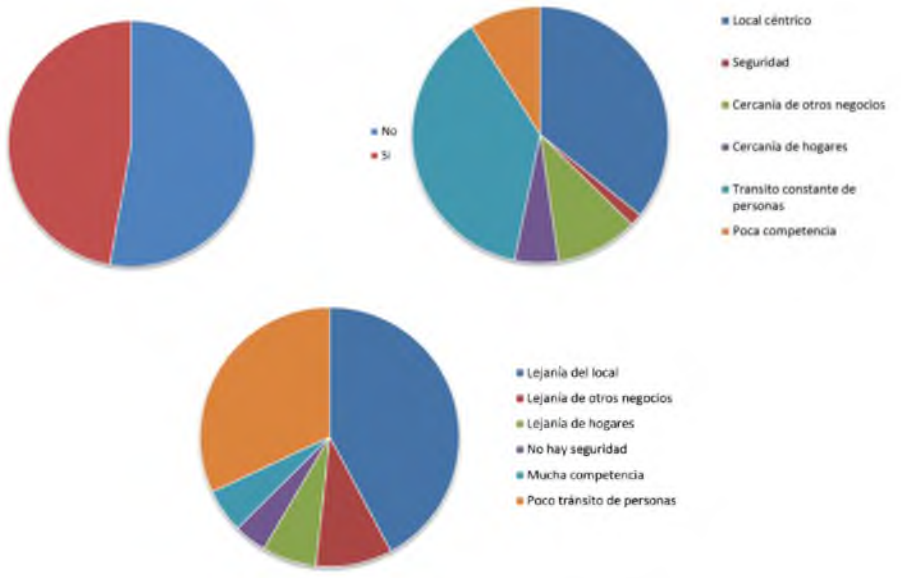


Figura 4.4 Gráficas generadas de una pregunta con tres respuestas

b) Gráficas por cédula de empresa. Esta opción generará las gráficas correspondientes para la información recabada de todas las cédulas de empresas. Para esta sección existen diversas opciones que permitirán obtener las gráficas, las cuales se mencionan a continuación:

1) Gráficas por colonia. Con esta opción solo se calcularán los datos para las encuestas realizadas a las empresas ubicadas en la colonia escogida.
2) Gráficas por giro. Con esta opción solo se calcularán los datos para las encuestas que pertenezcan al giro seleccionado. Esta opción también realizará gráficas para los datos del entrevistado.
3) Gráficas de anexos. Esta opción permitirá al usuario generar las gráficas de todas las preguntas que representen los anexos, tanto el Anexo A, como el Anexo B por separado. Todas las preguntas de los anexos son cerradas y con opción de una sola respuesta, por lo que el criterio aplicado para ellas será el de una sola gráfica por pregunta.
4) Gráficas por encuesta completa. Esta opción generará las gráficas de todas las preguntas de la encuesta, sin incluir los anexos. Cada pregunta deberá ser representada por las gráficas necesarias que describen los datos obtenidos. También se podrán aplicar filtros en base a colonia y giro de las empresas, al igual que las gráficas por cédula de empresa. Este punto es muy importante debido a que contendrá la información completa de todas las encuestas y al contener también todas las gráficas, el documento debe quedar completamente legible al ser generado.
5) Gráfica MiPYME. Esta gráfica indicará la identificación de las MiPYMES, utilizando un criterio basado en la ley para el desarrollo de la competitividad de las micro, pequeñas y medianas empresas de acuerdo a la última reforma publicada el 21 de Enero de 2015.

Estratificación por Número de Trabajadores			
Sector/Tamaño	**Industria**	**Comercio**	**Servicios**
Micro	0-10	0-10	0-10
Pequeña	11-50	11-30	11-50
Mediana	51-250	31-100	51-100

Figura 4.5 Tabla de identificación por número de trabajadores

Es importante mencionar que debido a que será difícil y poco probable que los negocios proporcionen la información correspondiente a sus ingresos, no

es recomendable estratificar en función de esta variable por lo que se recomienda solo considerar el número de empleados utilizando exclusivamente la estratificación presentada en la Figura 4.5. Cada opción generará un solo archivo en formato *xlsx* y se descargará automáticamente en el equipo del usuario.

4.1.3 Directorio de Empresas

Al finalizar el proceso de captura de las encuestas el sistema debe permitir visualizar todas las cédulas de los negocios encuestados a manera de un directorio, que servirá en un futuro para clasificar a las empresas y sus tipos de giro específico. Al tener este directorio se facilitará a todas aquellas personas o empresas que requieran de algún servicio, encontrar una empresa en la región que pueda realizar las actividades que se requieran. El directorio debe cumplir con las siguientes características:

- La información deberá ser mostrada a manera de tablas, con las siguientes columnas para cada cédula de encuesta: Nombre y/o razón social, Manzana, Dirección, Teléfono, Tipo de Empresa (Persona física, Sociedad Anónima, Sociedad Civil, Sociedad Cooperativa) y Forma de Operar (Independiente, Matriz, Sucursal, Distribuidor, Franquicia).
- Con el fin de facilitar la navegación entre las encuestas se debe disponer de una opción para mostrar las cédulas según lo seleccionado:
    - Mostrar por sector. Existen tres sectores: Industrial, Comercial y de Servicios. Al seleccionar cualquiera de los sectores, se deben mostrar sólo registros de cédulas que pertenezcan a ese sector.
    - Mostrar por tipo de negocio. Esta opción dividirá todas las cédulas en tipo de negocio utilizando la tabla de la Figura 4.5 y las mostrará según sean Micro, Pequeñas, o Medianas empresas. Las dos selecciones anteriores se especifican por separado o bien, se pueden mezclar las dos y mostrar un registro de cédulas más detallado. Por ejemplo, seleccionar Sector Industrial en el primer filtro y Micro Empresas en el segundo, con lo cual se obtendrán todos los registros de cédulas que pertenezcan al sector Industrial y que además sean Micro Empresas. Cada opción de consulta debe permitir generar un archivo PDF respetando los filtros seleccionados. El archivo

PDF generado se debe poder descargar al equipo del usuario.

Estas son las características principales que debe tener el sistema MiPYME-CAEO, teniendo en cuenta que según las necesidades del cliente, las características cambian, o bien se agregan nuevas.

4.2 Fase 2.- Colaboración

Una vez concluida la fase anterior se pasó a la fase de colaboración en donde se describe el proceso de desarrollo del sistema MiPYME-CAEO, tomando en cuenta las características principales descritas por el CAEO y las herramientas utilizadas así como su configuración y uso.

4.2.1 Iteración: Base de Datos

Como SGBD se eligió MySQL, que cuenta con diversas características y bondades, y facilitarán el desarrollo de la BD. Utilizando la herramienta MySQL *Workbench (v6.3)*, se modeló una BD de forma gráfica, con las características necesarias para permitir almacenar la información de las encuestas y que al mismo tiempo facilitara la obtención de la información para realizar las operaciones pertinentes sobre los registros.

Las tablas que conforman la estructura de la BD se realizaron conforme a la estructura del cuestionario. El cuestionario se encuentra formado por una Cédula de identificación del negocio (CIN), datos del entrevistado, noventa y cinco preguntas dicotómicas, de opción múltiple y abiertas, de las áreas de tecnologías de la información, mercadotecnia, administración y finanzas, una guía de observaciones del encuestador, dos anexos, uno de sesenta y siete preguntas, y otro de veintidós preguntas. En la Figura 4.7 se presenta la estructura final de la BD. Cabe mencionar que las tablas *auth_user, auth_user_groups, auth_user_user_permissions, auth_group, auth_group_permissions, auth_permission, django_session, django_content_type, django_admin_log,* son generadas automáticamente por Django al crear la estructura de la BD en las cuales se guardan datos de sesión de usuarios, permisos de usuarios, grupos de usuarios, entre otros. De las tablas generadas automáticamente, se utilizó la tabla *auth_user* para almacenar los usuarios que tendrían accesos al sistema. En la Figura 4.6

se presenta un diccionario de datos para describir cada una de las tablas utilizadas, sus campos, tipo de dato, tamaño y la relación con otra(s) tabla(s).

Tabla	Campo	Tipo	Tamaño	Tabla Relación	Descripción
colonia	id	INT	11		Tabla con registro de Colonias.
	nombre	VARCHAR	25		
	cp	INT	11		
giro	id	INT	11		Tabla con registro de Giros de empresas.
	giro	VARCHAR	10		
	tipo	VARCHAR	50		
	detalle	VARCHAR	50		
entrevistado	id	INT	11		Tabla con registros de los entrevistados.
	puesto	VARCHAR	25		
	genero	VARCHAR	10		
	antiguedad	INT	11		
	grado_academico	VARCHAR	15		
negocio	id	INT	11		Tabla donde se registra la información de los negocios entrevistados.
	nombre	VARCHAR	50		
	calle	VARCHAR	50		
	no	INT	11		
	colonia_id	FOREIGN		colonia	
	entre_calle1	VARCHAR	50		
	entre_calle2	VARCHAR	50		
	antiguedad	INT	11		
	telefono	INT	11		
	total_empleados	INT	11		
	total_mujeres	INT	11		
	total_hombres	INT	11		
	giro_id	FOREIGN		giro	
	tipo_empresa	VARCHAR	25		
	operacion	VARCHAR	50		
	e_mail	VARCHAR	75		
respuesta_tipo1	id	INT	11		Tabla con las respuestas de tipo 1
	respuesta	VARCHAR	230		
respuesta_tipo2	id	INT	11		Tabla con las respuestas de tipo 2
	respuesta	VARCHAR	100		
	campo_extra	VARCHAR	100		
respuesta_tipo3	id	INT	11		Tabla con las respuestas de tipo 3
	respuesta	VARCHAR	100		
	campo_extra1	VARCHAR	100		
	campo_extra2	VARCHAR	100		
pregunta	id	INT	11		Tabla que contiene todas las preguntas de la encuesta, incluidos anexos y guia de encuestador
	no_pregunta	INT	11		
	pregunta	VARCHAR	200		
	pertenece	VARCHAR	25		

Tabla	Campo	Tipo	Tamaño	Tabla Relación	Descripción
encuesta	id	INT	11		Tabla principal donde se almacenan los datos correspondientes a una encuesta, tales como el negocio entrevistado, el encuestador, entrevistado, entre otros.
	fecha	DATETIME			
	manzana	INT	11		
	encuestador_id	FOREIGN		auth_user	
	folio	VARCHAR	10		
	negocio_id	FOREIGN		negocio	
	entrevistado_id	FOREIGN		entrevistado	
	observaciones	LONGTEXT			
encuesta_pregunta	encuesta_id	FOREIGN		encuesta	Tabla donde se almacenan las relaciones entre preguntas y encuestas.
	pregunta_id	FOREIGN		pregunta	
pregunta_rtipo1	id	INT	11		Tabla que almacena las respuestas de tipo 1 almacenadas para la pregunta respondida de la encuesta.
	encuesta_preguntas_id	FOREIGN		encuesta_pregunta	
	respuestatipo1_id	FOREIGN		respuesta_tipo1	
pregunta_rtipo2	id	INT	11		Tabla que almacena las respuestas de tipo 2 almacenadas para la pregunta respondida de la encuesta.
	encuesta_preguntas_id	FOREIGN		encuesta_pregunta	
	respuestatipo2_id	FOREIGN		respuesta_tipo2	
pregunta_rtipo3	id	INT	11		Tabla que almacena las respuestas de tipo 3 almacenadas para la pregunta respondida de la encuesta.
	encuesta_preguntas_id	FOREIGN		encuesta_pregunta	
	respuestatipo3_id	FOREIGN		respuesta_tipo3	
posible_rtipo1	id	INT	11		Tabla con la relación entre una pregunta y sus posibles respuestas tipo 1.
	pregunta_id	FOREIGN		pregunta	
	respuestatipo1_id	FOREIGN		respuesta_tipo1	
posible_rtipo2	id	INT	11		Tabla con la relación entre una pregunta y sus posibles respuestas tipo 2.
	pregunta_id	FOREIGN		pregunta	
	respuestatipo2_id	FOREIGN		respuesta_tipo2	
posible_rtipo3	id	INT	11		Tabla con la relación entre una pregunta y sus posibles respuestas tipo 3.
	pregunta_id	FOREIGN		pregunta	
	respuestatipo3_id	FOREIGN		respuesta_tipo3	
auth_user	id	INT	11		Tabla donde se almacenan los usuarios que tendrán acceso a la encuesta, encuestadores, administradores, entre otros.
	password	VARCHAR	128		
	last_login	DATETIME			
	is_superuser	TINYINT	1		
	username	VARCHAR	30		
	first_name	VARCHAR	30		
	last_name	VARCHAR	30		
	email	VARCHAR	75		
	is_staff	TINYINT	1		
	is_active	TINYINT	1		
	date_joined	DATETIME			

Figura 4.6. Diccionario de Datos

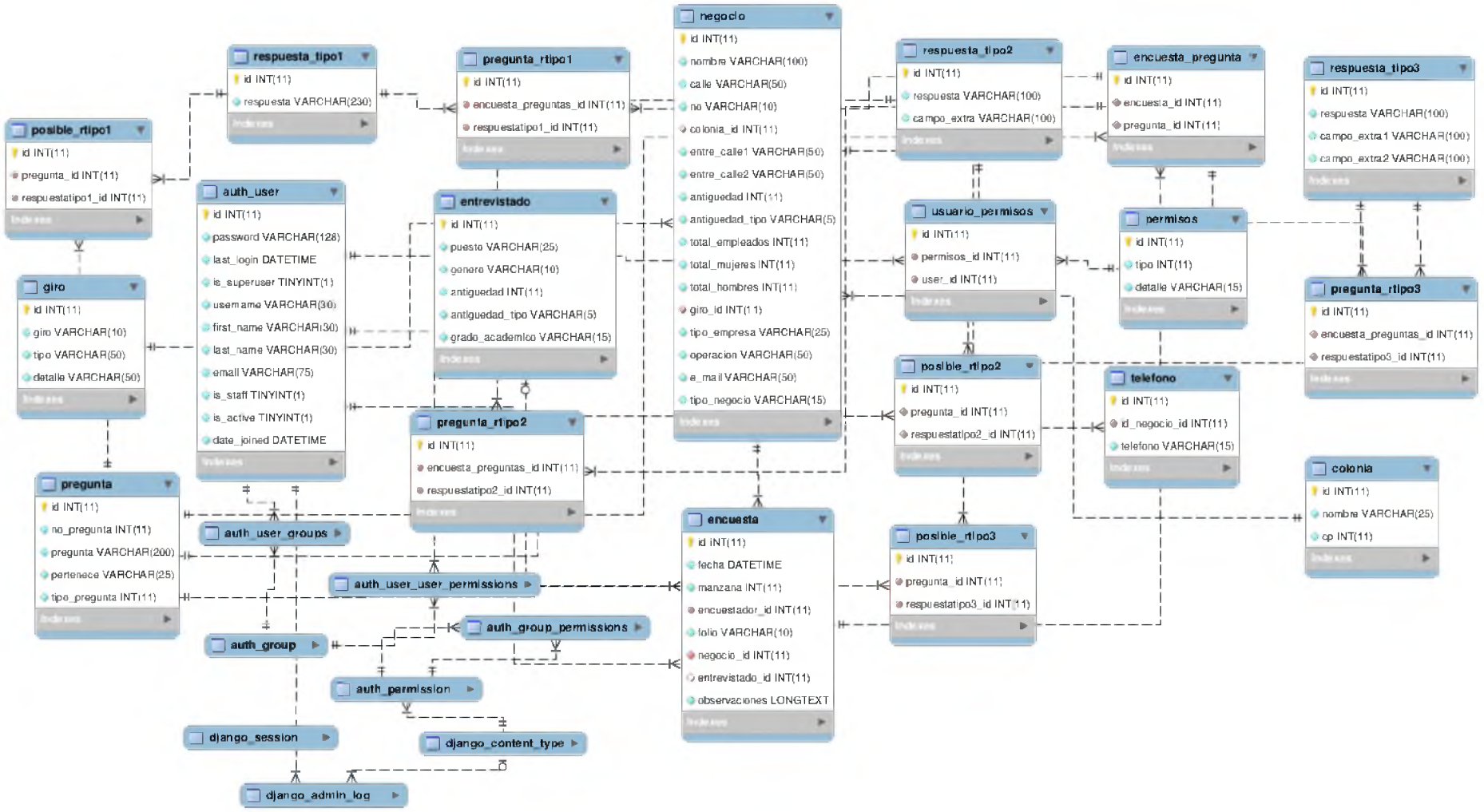


Figura 4.7 Estructura de la BD

Con la estructura de la base de datos ya generada se procede a probar la misma y verificar que las funcionalidades sean las esperadas. Posteriormente se genera la BD utilizando la misma estructura creada en *Workbench*, pero ahora a partir de los modelos de Django haciendo uso de la tecnología ORM (Modelo Objeto-Relacional) con los siguientes pasos:

1. Lo primero es crear el proyecto en Django, para esto se requiere tener instalado Django en el sistema. En este caso, el sistema operativo es Xubuntu 15.04. Se requieren de las siguientes librerías y paquetes: *libmysqlclient-dev*, *python-dev* y *libevent-dev*, *virtualenv*, previamente instalados en el sistema. Para instalar Django, se creó un ambiente virtual, en el que se instalaron todos los paquetes necesarios, sin modificar el sistema principal. Con la herramienta *VirtualEnv* se crea el ambiente virtual utilizando el comando "virtualenv pyme".
2. Una vez creado, se ejecuta el ambiente virtual con el comando: "*source pyme/bin/activate*"
3. Con esto, la terminal ya se encuentra trabajando con los paquetes *Python* del ambiente virtual y se prosigue a instalar los paquetes necesarios para correr el sistema MiPYME-CAEO. A partir de este punto, todos los comandos para instalar paquetes, se deberán ejecutar dentro del ambiente virtual. Para instalar Django se ejecuta el comando en la terminal: "*pip install django==1.6*". La parte de *"==1.6"*, indica la versión de Django que se desea instalar. Para evitar errores de compatibilidad, se definirá esa versión. De otra manera, si no se especifica una versión, se instala la más reciente.
4. Para crear el proyecto se ejecuta en la terminal el siguiente código: "*django-admin.py startproject pyme_v1*"
5. Este comando generará la estructura del proyecto de la siguiente manera:

```
pyme_v1/
    manage.py
    pyme_v1/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Donde los archivos son:

- ***manage.py***. Es una utilidad para línea de comandos, que permite la interacción de diversas maneras con el proyecto. Por lo general se utiliza en el entorno de

desarrollo ya que se verifican datos de los modelos sin necesidad de entrar a un SGBD, de igual manera se despliega un ambiente de pruebas sin la necesidad de utilizar un servidor de producción. Este ambiente de pruebas se ejecuta con el siguiente código: "*python manage.py runserver*". Se desplegará un ambiente de pruebas accesible desde la dirección *http://127.0.0.1:8000/* en cualquier explorador Web del sistema.

- ***__init.__py***. Este es un archivo vacío que cumple la funcionalidad de indicarle a *Python* que el directorio donde se encuentra debe ser agregado como un paquete de *Python*. Esta es una metodología que utiliza *Python* llamada "nombres de módulos con puntos", la cual permite la relación entre los sub-módulos dentro del paquete.
- ***settings.py***. Este archivo contiene todas las configuraciones del proyecto como las variables de la base de datos para realizar la conexión. Por default, Django genera una BD utilizando SQLite, pero para el sistema del CAEO se utilizará MySQL, por lo que se deben cambiar las variables de la BD a las de MySQL. En el siguiente código se describen las variables configuradas para utilizar una BD de MySQL:

```
# Database
# https://docs.djangoproject.com/en/1.6/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql', # Add
'postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.
        'NAME': 'pymedb',          # Or path to database file if using
sqlite3.
        'USER': 'pyme',                      # Not used with sqlite3.
        'PASSWORD': 'admincaeo',                  # Not used with
sqlite3.
        'HOST': '',                      # Set to empty string for
localhost. Not used with sqlite3.
        'PORT': '',                      # Set to empty string for
default. Not used with sqlite3.
    }
}
```

Dónde:

- ✓ ***ENGINE***. Define el motor de BD que se utilizará.
- ✓ ***NAME***. Es el nombre de la BD que se va a utilizar.
- ✓ ***USER***. El usuario que tiene acceso a la BD.
- ✓ ***PASSWORD***. Contraseña de acceso para la BD.

- ✓ ***HOST***. La dirección en la cual se encuentra la BD, donde al dejar vacío el campo, se indica a Django que la BD se encuentra en el mismo sistema que estará corriendo el proyecto. En este campo se debe indicar la dirección en caso que la BD se acceda de manera remota.
- ✓ ***PORT.*** De la misma forma que *HOST*, se deja vacío para indicar que el puerto es el default utilizado por MySQL. Se indica el puerto de acceso a la máquina remota si es el caso.

Se requiere de instalar el paquete "*MySQL-python==1.2.5*", de la misma forma que se instaló Django, "*pip install MySQL-python==1.2.5*", para que se pueda tener acceso a las bases de datos de MySQL. En este punto ya se debe tener una BD creada en MySQL.

- ***urls.py***. Este archivo es un despachador de urls, que puede describirse como el contenido del sitio. Es un módulo escrito también en *Python*, y define los procedimientos a seguir según las urls solicitadas al sistema, cuando se navega en la página del proyecto en un explorador. El funcionamiento de este archivo se describirá más adelante.
- ***wsgi.py***. WSGI (*Web Server Gateway Interface*), es un archivo de configuración para servidores, que permite ejecutar la aplicación de Django en ambientes de producción. Este archivo será descrito cuando se configure para correr con el servidor Apache.

6. Una vez que se tiene el proyecto, se procede a generar la aplicación o aplicaciones. Para crear una aplicación se ejecuta, mediante la consola de comandos, el siguiente comando: "*python manage.py startapp encuestas*".
7. Con este comando se agregará la carpeta correspondiente a la aplicación "encuestas" y sus respectivos archivos de configuración principal.

```
encuestas/
    __init__.py
    admin.py
    models.py
    tests.py
    views.py
```

Dónde:

- **__init__.py**. Este es un archivo vacío que cumple la funcionalidad de indicarle a *Python* que el directorio donde se encuentra debe ser agregado como un paquete de *Python*. Esta es una metodología que utiliza *Python* llamada "nombres de módulos con puntos", la cual permite la relación entre los sub-módulos dentro del paquete.
- **admin.py**. Este archivo funciona con Django, y permite agregar información mediante modelos, utilizando una interfaz *Web* predefinida. Esta particular herramienta permite agregar información a la BD, sin la necesidad de generar una página específica o realizar rutinas de volcado de información.
- **models.py**. Este archivo contiene todos los modelos de la aplicación.
- **tests.py**. Este archivo funciona a manera de pruebas, se generan rutinas de prueba para los modelos, realizando diversas actividades. Esto con el fin de evitar errores en las rutinas que se utilicen a partir de los modelos.
- **views.py**. En este archivo se alojan las funciones que vaya a realizar el sistema, con respecto al re-direccionamiento de URLS y diversas otras rutinas que tengan que ver con la aplicación.

Una vez que se tiene la estructura de la aplicación, se generan los modelos. Un modelo de Django es una estructura escrita en código *Python* denominada como Clase, que será convertida en una tabla de BD. Los campos que conforman la estructura serán representados de igual manera en la tabla de BD, cada campo debe especificar su tipo de dato, tamaño máximo de caracteres, entre otros aspectos. En el siguiente fragmento de código se describe el modelo "Colonia", representado como una clase de *Python*, formada por dos campos:

```
class Colonia(models.Model):
    nombre = models.CharField(max_length=25)
    cp = models.IntegerField()

    class Meta:
        db_table = 'colonia'
    ordering = ['id']

    def __unicode__(self):
        return self.nombre
```

En donde:

1. "nombre", de tipo *CharField* (equivalente a *VARCHAR* de MySQL) con límite de 25 caracteres.

2. "cp", de tipo *IntegerField* (equivalente a *INT* de MySQL) sin límite de dígitos. Los dos campos serán columnas de la tabla en la BD.

Dentro del modelo "Colonia" podemos observar una clase interna de *Python* llamada "Meta", la cual contiene dos campos:

1. "***db_table***", que indica el nombre que tendrá la tabla en la BD. Por default Django pone un nombre a la tabla si no se especifica.
2. "***ordering***", este campo indicará el orden los datos obtenidos después de ejecutar un *query* con los modelos de Django. Este campo recibe una tupla, por lo que se indican más campos para el ordenamiento si es necesario. Este campo es opcional y por default se especifica recibir los datos ordenados por medio del ID del modelo. El campo ID se agrega a cada modelo, aunque el diseñador no lo especifique al momento de agregar los campos.

La función *"__unicode__"* (__unicode__ se utiliza para *Python* 2.7, __str__ se utiliza para *Python* 3.0), indica el campo por default que será visualizado cada que se realice un *query*. En el caso del modelo "Colonia", el campo que se mostrará es "nombre".

En la estructura generada con *Workbench* se planteó el uso de relaciones 1-N y M-N, por ejemplo, en la relación de una pregunta con sus posibles respuestas, teniendo en cuenta que una pregunta o varias preguntas, tienen de una a varias respuestas. Django en su versión 1.6 no maneja una configuración específica para la generación automática de las tablas que se crean en una relación M-N, por lo que el desarrollador puede generar las tablas por cuenta propia y después indica la multi-relación en los modelos que lo requieran. Las relaciones con modelos de Django son:

- Relación 1-1.- Una relación 1-1 requiere de dos modelos, donde uno de ellos contiene un campo del tipo *OneToOneField*, que recibe como parámetro el modelo con el cual formará la relación 1-1. En el siguiente código se ejemplifica la relación 1-1 utilizando dos modelos, "Lugar" y "Restaurant", donde "Lugar" representa la ubicación de un restaurant y "Restaurant" contiene un campo con relación 1-1, que apunta al modelo "Lugar". Como dato extra, un segundo parámetro en el campo lugar, del modelo "Restaurant", denominado "*primary_key*" igual con *True*, define ese campo como la llave primaria del modelo y por lo tanto en la tabla de la base de datos. Todos los modelos de Django definen por default una llave primaria ID si no se especifica en el modelo por el programador.

```
class Lugar(models.Model):
    nombre = models.CharField(max_length=50)
    direccion = models.CharField(max_length=80)

    # En Python 3: def __str__(self):
    def __unicode__(self):
        return u"%s el lugar" % self.nombre

class Restaurant(models.Model):
    lugar = models.OneToOneField(Lugar, primary_key=True)
    sirve_hot_dogs = models.BooleanField()
    sirve_pizza = models.BooleanField()

    # En Python 3: def __str__(self):
    def __unicode__(self):
        return u"%s el restaurant" % self.lugar.nombre
```

- Relación 1-N.- De igual manera se requieren dos modelos para la relación 1-N o "Uno a Muchos". El tipo de campo que define la relación es del tipo *ForeignKey*, que recibe como parámetro la clase del modelo al que será relacionado. En el siguiente código se tienen dos modelos, "Encuesta" que representa el modelo principal del sistema CAEO, formado por siete campos, de los cuales tres son *ForeignKey*.

```
class Encuesta(models.Model):
    fecha = models.DateTimeField()
    manzana = models.IntegerField()
    encuestador = models.ForeignKey(User)
    folio = models.CharField(max_length=10)
    negocio = models.ForeignKey(Negocio)
    entrevistado = models.ForeignKey(Entrevistado, blank=True, null=True)
    observaciones = models.TextField()

    class Meta:
        db_table = 'encuesta'


class Entrevistado(models.Model):
    puesto = models.CharField(max_length=25)
    genero = models.CharField(max_length=10)
    antiguedad = models.IntegerField()
    antiguedad_tipo = models.CharField(max_length=5)
    grado_academico = models.CharField(max_length=15)

    class Meta:
        db_table = 'entrevistado'
```

Se tomará como referencia el modelo "Entrevistado", este contiene dos parámetros que determinarán la propiedad de ingresar una encuesta sin que se haya entrevistado a un empleado o dueño de la empresa. *"blank"* define si el campo será

requerido al momento de generar un registro de ese modelo, por lo que al definirlo como *True*, el campo no es requerido. Por otra parte, *"null"* indica la opción de un campo para adoptar la propiedad *NULL*. En este caso el campo entrevistado es una llave foránea hacia un registro del modelo "Entrevistado", por lo que al quedar vacío, no se le da un valor entero o de cadena, así que se quedará con valor *NULL* en esos casos. En el caso de los tipos de datos *CharField* e *IntegerField*, se define un valor *default* en dado caso que no se especifique uno al momento de generar el registro, esto con el parámetro *"default"*.

- Relación M-N.- El campo que indica una relación "Muchos a Muchos" es *ManyToManyField*, el cual recibe dos parámetros, el primero indicando el otro modelo de la relación y el segundo que indica el nombre de la tabla en la BD. Esta relación genera automáticamente una tabla en la BD, pero no genera el modelo de la relación, sino que se accede mediante el modelo donde se agrega la instancia de la relación M-N, como se muestra en la Figura 4.8.

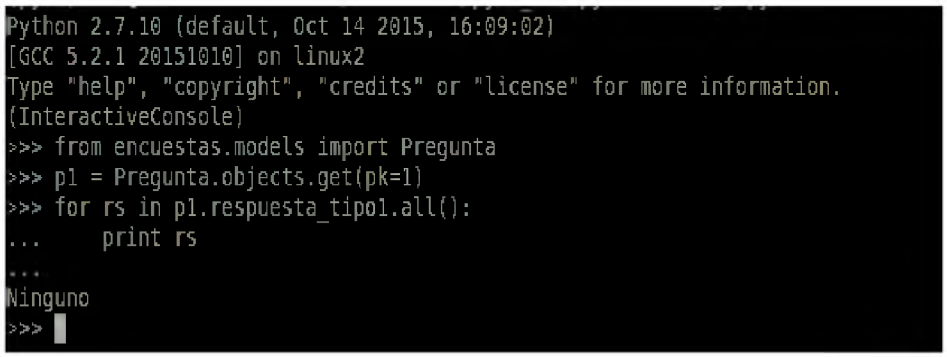


Figura 4.8 Acceso a las relaciones M-N utilizando modelos de Django

En el siguiente código se describen los modelos utilizados en el ejemplo de la Figura 4.8.

```
class Pregunta(models.Model):
    no_pregunta = models.IntegerField()
    pregunta = models.CharField(max_length=200)
    pertenece = models.CharField(max_length=25)
    tipo_pregunta = models.IntegerField()
    respuesta_tipo1 = models.ManyToManyField(RespuestaTipo1, db_table=
'posible_rtipo1')
    respuesta_tipo2 = models.ManyToManyField(RespuestaTipo2, db_table=
'posible_rtipo2')
    respuesta_tipo3 = models.ManyToManyField(RespuestaTipo3, db_table=
'posible_rtipo3')

    class Meta:
        db_table = 'pregunta'
        ordering = ['no_pregunta', 'pertenece']

    def __unicode__(self):
        return '%d' % self.no_pregunta + '.-' + self.pregunta


class RespuestaTipo1(models.Model):
    respuesta = models.CharField(max_length=230)

    class Meta:
        db_table = 'respuesta_tipo1'

    def __unicode__(self):
        return self.respuesta
```

El modelo "Pregunta", tiene tres relaciones M-N, hacia los modelos "RespuesaTipo1", "RespuestaTipo2" y"RespuestaTipo3" de los cuales se generarán sus respectivas tablas en la base de datos y se tendrá acceso a ellas mediante el modelo "Pregunta". La ventaja de esa configuración es que permite reutilizar respuestas en diversas preguntas e inclusive agregar más respuestas en un futuro y asignarlas de manera sencilla a la pregunta o preguntas correspondientes. Una vez creados todos los modelos, se procede a sincronizarlos con la base de datos. Para esto se requiere de un *"schema"* ya creado en la base de datos. Es muy importante utilizar codificación UTF-8 en la base de datos y codificar los archivos de *Python* con UTF-8 ya que por *default Python* no reconoce caracteres latinos. Antes de sincronizar la BD con los modelos se debe ejecutar el comando: "*python manage.py sql encuestas*". El anterior comando genera la estructura de las tablas que serán agregadas en la BD, como se muestra en la Figura 4.9 de manera que se verifica si existe algún problema, permitiendo corregirlo antes de sincronizar.

```
(pyme)lobo@lobo-VirtualBox:~/Documents/pyme_v1$ python manage.py sql encuestas
BEGIN;
CREATE TABLE `colonia` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `nombre` varchar(25) NOT NULL,
    `cp` integer NOT NULL
)
;
CREATE TABLE `giro` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `giro` varchar(10) NOT NULL,
    `tipo` varchar(50) NOT NULL,
    `detalle` varchar(50) NOT NULL
)
;
CREATE TABLE `entrevistado` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `puesto` varchar(25) NOT NULL,
    `genero` varchar(10) NOT NULL,
    `antiguedad` integer NOT NULL,
    `antiguedad_tipo` varchar(5) NOT NULL,
    `grado_academico` varchar(15) NOT NULL
)
;
CREATE TABLE `usuario_permisos` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `permisos_id` integer NOT NULL,
    `user_id` integer NOT NULL,
    UNIQUE (`permisos_id`, `user_id`)
)
```

Figura 4.9 SQL de los modelos de Django

Posteriormente, el comando para sincronizar es el siguiente: "*python manage.py syncdb*". Con esto se generarán las tablas en la BD y podrán ser accesibles mediante los modelos de Django. El paso siguiente es llenarla con los valores iniciales que requiere la encuesta, como las Preguntas, Respuestas, Giros, entre otros datos.

4.2.2 Iteración: Acceso e inserción de información con modelos de Django

El proceso para entrar en un ambiente de pruebas de Django, llamado "*Shell*", y ejecutar *queries* para obtener los datos de la BD, se observa en la Figura 4.10, en donde como primer paso se entra al ambiente virtual generado con la herramienta *VirtualEnv*. *VirtualEnv* es una herramienta que genera ambientes aislados de *Python*, esto quiere decir que se puede crear un ambiente *Python* completamente independiente al del sistema operativo. La ventaja de generar un ambiente virtual es tener aislado el sistema, de esta forma se pueden tener múltiples proyectos, cada uno utilizando su propia

configuración, sin la necesidad de modificar la instalación *python* principal del sistema operativo.

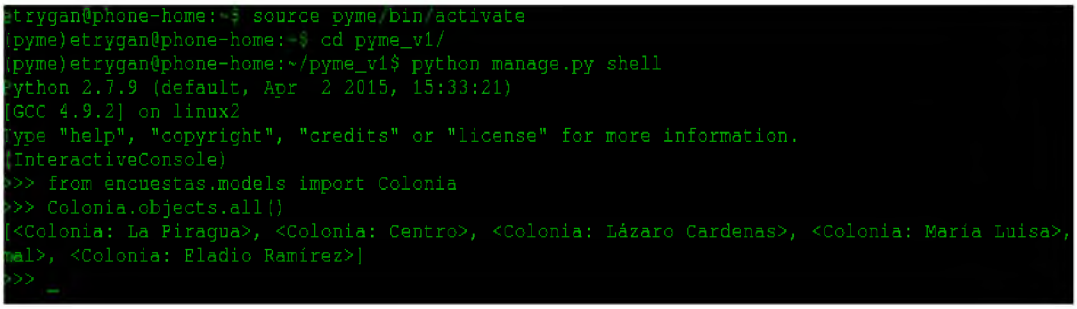


Figura 4.10 Imprimiendo el resultado de un *query* con los modelos de Django

Se tienen pre-establecidas respuestas para las preguntas de la encuesta, para los tipos de giros se cuenta con una lista, así como diversas variables que se deben ingresar a la base de datos desde el inicio del sistema. Para ingresar los datos al sistema se generó un script en *Python*, que lee información almacenada en archivos CSV (*comma separated values*), que es un formato libre para representar datos a manera de tablas, para después guardarla en los modelos correspondientes. El siguiente código muestra el ejemplo de llenado de dos modelos, Giros y Preguntas.

```
def add_preguntas():
    with open('populate_pyme/preguntas.csv', 'rb') as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"')
        print "Adding Preguntas... "
        for row in reader:
            (No, pregunta, pertenece) = row
            Pregunta.objects.get_or_create(no_pregunta=No,
            pregunta=pregunta, pertenece=pertenece)
        print "Done"


def add_giros():
    with open('populate_pyme/giros_New.csv', 'rb') as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"')
        print "Adding Giros... "
        for row in reader:
            (giro, tipo, detalle) = row
            Giro.objects.get_or_create(giro=giro, tipo=tipo,
detalle=detalle)
        print "Done"
```

```
class Giro(models.Model):
    giro = models.CharField(max_length=10)
    tipo = models.CharField(max_length=50)
    detalle = models.CharField(max_length=50)

    class Meta:
        db_table = 'giro'
    ordering = ['id']

    def __unicode__(self):
        return self.detalle


class Pregunta(models.Model):
    no_pregunta = models.IntegerField()
    pregunta = models.CharField(max_length=200)
    pertenece = models.CharField(max_length=25)
    tipo_pregunta = models.IntegerField()
    respuesta_tipo1 = models.ManyToManyField(RespuestaTipo1, db_table=
'posible_rtipo1')
    respuesta_tipo2 = models.ManyToManyField(RespuestaTipo2, db_table=
'posible_rtipo2')
    respuesta_tipo3 = models.ManyToManyField(RespuestaTipo3, db_table=
'posible_rtipo3')

    class Meta:
        db_table = 'pregunta'
        ordering = ['no_pregunta', 'pertenece']

    def __unicode__(self):
        return '%d' % self.no_pregunta + '.-' + self.pregunta
```

El proceso para las dos funciones, lee el archivo CSV y los guarda uno por uno en los modelos, empleando la función "*get_or_create*", la cual se recomienda utilizar para guardar grandes cantidades de datos o simplemente para evitar almacenar datos duplicados. El código anterior es parte de un archivo de llenado masivo, creado específicamente para ingresar toda la información requerida que debe contener el sistema para poder mostrar información, como las preguntas, las posibles respuestas, giros, colonias, entre otros datos. La función "*get_or_create*" primero trata de obtener el registro, si detecta que el registro no existe, lo crea. Como parámetros se deben pasar los campos que forman el modelo, por ejemplo, para el modelo "Giro" se deben indicar los valores de los campos "giro", "tipo", y "detalle". En el caso del modelo "Pregunta", los campos que representan la relación M-N, no se asocian sin antes haber guardado los datos principales. Esto es, primero guardar los datos principales del modelo, "no_pregunta", "pregunta" y "pertenece". La función ""*get_or_create*" automáticamente salva el modelo y después de esto se le agregan los campos de relación

M-N. El siguiente código describe como agregar las relaciones M-N después de respaldar el modelo.

```
class Publicacion(models.Model):
    titulo = models.CharField(max_length=30)

    def __unicode__(self):              # __str__ on Python 3
        return self.titulo

    class Meta:
        ordering = ('titulo',)

class Articulo(models.Model):
    encabezado = models.CharField(max_length=100)
    publicaciones = models.ManyToManyField(Publicacion)

    def __unicode__(self):              # __str__ on Python 3
        return self.encabezado

    class Meta:
        ordering = ('encabezado',)

p1 = Publicacion(titulo='The Python Journal')
p1.save()

a1 = Articulo(encabezado='Django lets you build Web apps easily')
a1.save()

a1.publicaciones.add(p1)
```

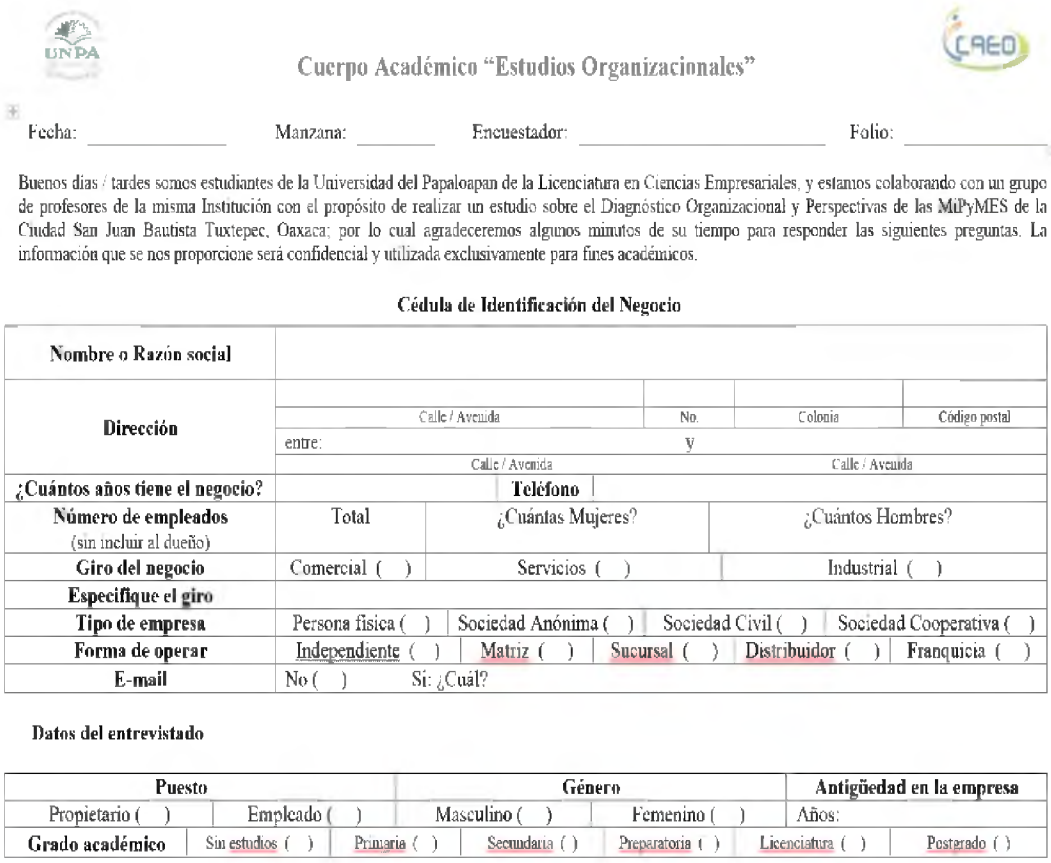
El modelo "Articulo" contiene un campo M-N en relación con el modelo "Publicacion", en el ejemplo anterior primero se agrega una publicación y se salva en el modelo. Después al guardar un artículo, se agrega primero el campo "encabezado", se salva el artículo y después se agrega la publicación al artículo con el comando "*add*" [39].

4.2.3 Iteración: Generación de Templates

Para la creación de la interfaz de usuario, se utilizaron *templates* de Django. Los *templates* son archivos HTML, con la opción de utilizar comandos de Django para realizar ciertas tareas, como la manipulación directa en el *template* de los modelos, toma de decisiones con sentencias "*IF-ELSE*", generación de *tags* HTML con ciclos *FOR* y una de las principales características, herencia de *templates*.

Las primeras interfaces en ser creadas son las que contienen los formularios de cédula de empresa y las secciones de preguntas. Se siguió el mismo diseño presentado

por el CAEO en el formato impreso de la encuesta para la cédula de las empresas, como se muestra en la Figura 4.11.



Cuerpo Académico "Estudios Organizacionales"

Fecha: _____ Manzana: _____ Encuestador: _____ Folio: _____

Buenos días / tardes somos estudiantes de la Universidad del Papaloapan de la Licenciatura en Ciencias Empresariales, y estamos colaborando con un grupo de profesores de la misma Institución con el propósito de realizar un estudio sobre el Diagnóstico Organizacional y Perspectivas de las MiPyMES de la Ciudad San Juan Bautista Tuxtepec, Oaxaca; por lo cual agradeceremos algunos minutos de su tiempo para responder las siguientes preguntas. La información que se nos proporcione será confidencial y utilizada exclusivamente para fines académicos.

Cédula de Identificación del Negocio

Nombre o Razón social				
Dirección	Calle / Avenida	No.	Colonia	Código postal
	entre: Calle / Avenida	y Calle / Avenida		
¿Cuántos años tiene el negocio?	**Teléfono**			
Número de empleados (sin incluir al dueño)	Total	¿Cuántas Mujeres?	¿Cuántos Hombres?	
Giro del negocio	Comercial ( )	Servicios ( )	Industrial ( )	
Especifique el giro				
Tipo de empresa	Persona física ( )	Sociedad Anónima ( )	Sociedad Civil ( )	Sociedad Cooperativa ( )
Forma de operar	Independiente ( )	Matriz ( )	Sucursal ( )	Distribuidor ( ) Franquicia ( )
E-mail	No ( )	Sí: ¿Cuál?		

Datos del entrevistado

Puesto		**Género**		**Antigüedad en la empresa**
Propietario ( )	Empleado ( )	Masculino ( )	Femenino ( )	Años:
Grado académico	Sin estudios ( )	Primaria ( )	Secundaria ( )	Preparatoria ( ) Licenciatura ( ) Postgrado ( )

Figura 4.11 Cédula de identificación de la empresa

En el siguiente código se describe el ejemplo de un *template*, utilizando funcionalidades propias del *Framework*.

```
<table align="center">
                <tr>
                    <td> Fecha:
                    <input type="date" name="fecha" id="datePicker"
min="2014-01-01">
                    </td>
                    <td> Manzana*:{{form.manzana}} </td>
                </tr>
                <tr>
                    <td> Encuestador:
                    <input id="little" type="text" name="encuestador"
readonly value="{{ user }}">
                    </td>
                    <td> Folio*:{{form.folio}} </td>
                </tr>
            </table>
```

Como se puede ver en el código anterior, se imprimen las variables enviadas al *template*, indicando siempre la variable entre llaves dobles. En el siguiente código se presenta un ejemplo utilizando el ciclo *For* en los *templates*.

```
{% extends "encuestas/base.html" %}
{% block content %}
<h3>Anexo B</h3>
<form action="{% url 'encuestas:anexoB' %}" method="post">
    {% csrf_token %}
    <div style="display: table;">
        {% for pregunta in preguntas %}
        <div style="border:2px solid black; display: table-row;" >
            <div style="display: table-cell;">
                <ul>
                    <label> {{ pregunta.pk }}.-{{ pregunta.pregunta }}
</label>
                    <br />
                    {% for respuestas in
pregunta.respuesta_tipo1.all|dictsortreversed:"pk" %}
                    <input type="radio" align="right" name="{{ pregunta.pk
}}" value="{{ respuestas.pk }}"/>
                    <label > {{ respuestas }} </label>
                    {% endfor %}
                    <br /><br />
                </ul>
            </div>
        </div>
        {% endfor %}
    </div>
    <div>
        <input type="submit" value="Siguiente" />
    </div>
</form>
{% endblock %}
```

En este caso, el código Django para representar ciclos, sentencias o *meta-tags*, se debe encontrar entre una llave y un signo de porcentaje al inicio, un signo de porcentaje y una llave de cierre al final. En el ejemplo anterior, se le manda al *template* todas las preguntas que corresponden al Anexo B, se separa cada una de ellas en el ciclo *FOR* y en base a ello, se generan las etiquetas HTML y los valores de la pregunta que serán visualizados en el explorador. Todo ciclo, sentencia o meta-tag que se abra utilizando el lenguaje de Django, debe tener una etiqueta de cierre. En el caso del *FOR*, la etiqueta de cierre es "*endfor*". La primera etiqueta abierta al inicio del *template* con la palabra "*extends*", indica la herencia de Django con respecto a los *templates*. *Extends* "*encuestas/base.html*" indica que el *template* actual heredará las estructuras del *template base.html*. El uso principal en la herencia de *templates*, es para indicar menús de página y heredarlos a todos los *templates* que lo vayan a necesitar, esto evita que el código del menú se tenga que re-escribir en todos los demás *templates*, ahorrando una cantidad de tiempo importante al momento del diseño de la interfaz *Web*.

Las interfaces de los *templates* son dinámicas, utilizando *javascript* y *jQuery*, con el fin de tener una interfaz más fluida y amigable para el usuario. La finalidad de esto es tener una interfaz inteligente, que permita el llenado de las encuestas en poco tiempo. Las funcionalidades principales son: ocultar preguntas en base a previas respuestas obtenidas de otras preguntas y mostrar posibles respuestas de selección según una previa selección realizada. A continuación se presenta un ejemplo de funcionalidad escrita en *jQuery*:

```
$('#id_giros').on('change', function() {
        $.ajax({
            type : "POST",
            url : '/encuestas/change_sector/',
            data : {
                csrfmiddlewaretoken : $.cookie('csrftoken'),
                giro : $(this).val(),
            },
            success : function(data) {
                var giros = data.split('#');
                var secciones = giros[0].split('&');
                var seccion_select = $('#id_seccion').empty();
                var tipo_giros = giros[1].split('&');
                var tipo_giros_select = $('#id_tipo_giro').empty();
                for (var i = 0; i < secciones.length; i++) {
                    seccion_select.append('<option value="' + secciones[i] +
'">' + secciones[i] + '</option>');
                }
                for (var j = 0; j < tipo_giros.length; j++) {
                    tipo_giros_select.append('<option value="' +
tipo_giros[j] + '">' + tipo_giros[j] + '</option>');
                }
            },
            error : function(xhr, textStatus, errorThrown) {
                alert("Please report this error: " + errorThrown + xhr.status
+ xhr.responseText);
            }
        });
    });
```

El código *jQuery* anterior se utiliza para obtener los tipos de giro, por ejemplo al seleccionar los tipos de giro Industriales, la función *Ajax* anterior realiza el proceso de obtener los tipos de giro industriales por medio de consultas directamente al servidor. La función *Ajax* envía el giro seleccionado a la función "*change_sector*" la cual regresa una cadena concatenada con todos los tipos de giros pertenecientes al sector industrial. Con un proceso de división de cadenas por medio de caracteres especiales colocados intencionalmente en la cadena obtenida, se separan los tipos de giros y mediante la creación dinámica de código HTML, se generan las opciones de la etiqueta "*select*". En el siguiente ejemplo se describe la función "*change_sector*", en donde se observa la consulta hacia el modelo Giro, filtrando los resultados en base al giro seleccionado en el *template*.

```
def change_sector(request):
    if request.is_ajax():
        try:
            giro = request.POST['giro']
        except KeyError:
            return HttpResponse('Error')
        secciones = Giro.objects.all().filter(giro=giro)
        sc = '&'.join(['%s' % s['tipo'] for s in
secciones.values('tipo').annotate(total=Count('tipo'))])
        tipo_giros = Giro.objects.all().filter(giro=giro, tipo='Alimentos y
bebidas')
        tg = '&'.join(['%s' % s['detalle'] for s in
tipo_giros.values('detalle').annotate(total=Count('detalle'))])
        return HttpResponse((sc + '#' + tg))
    else:
        raise Http404
```

Todas las peticiones son gestionadas por el archivo *urls.py*, este archivo funciona como despachador de direcciones, que le indica al sistema la ruta que debe seguir, dependiendo de la petición realizada. En el siguiente código se describe el formulario del *template* de *Login*, para el sistema.

```
{% load static %}
{% block content %}
{% csrf_token %}
<form action="{% url 'login' %}" method="post">
    {% csrf_token %}
    <div  align="center">
        <img src="{% static 'img/logoUNPA.jpg' %}" width="160px"
height="190px" alt="UNPALOGO">
        <h3>Bienvenido al sistema MiPYME-CAEO</h3>
        <div id="login">
            {{ form.as_p }}
            <input type="submit" value="Entrar" />
        </div>
    </div>
    <div align="center">
        {{ message }}
    </div>
</form>
{% endblock %}
```

El código anterior, genera la interfaz de *login* para el sistema, como se muestra en la Figura 4.12, utilizando las cualidades de Django para generar archivos con poco código y además entendible. El formulario ejecuta una petición *POST* al sistema, el cual es dirigido por el sistema utilizando la ruta "*login*".

Figura 4.12 Interfaz de login para el sistema MiPYME-CAEO

El despachador de direcciones busca la dirección solicitada en el archivo *urls.py* y re-direcciona la petición. El siguiente código describe el archivo *urls.py* del proyecto principal y la línea para la petición de formulario de ingreso.

```
from django.conf.urls import patterns, include, url
from encuestas import views
from django.core.urlresolvers import reverse_lazy

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^$', views.login, name='login'),
    url(r'^logout/$', 'django.contrib.auth.views.logout',
          {
            "next_page" : reverse_lazy('login')
          }, name="logout"),
    url(r'^encuestas/', include('encuestas.urls', namespace="encues-
tas")),
)
```

Django recomienda que se tenga un archivo *urls.py* principal del proyecto y uno dentro de cada aplicación que conforme el proyecto. En este caso el código anterior representa el archivo *urls.py* del proyecto. La línea subrayada representa la URL de petición que deberá seguir el sistema para el *login* al sistema. El primer parámetro representa la expresión regular de la dirección, el segundo campo, indica el archivo y la función a la cual se deberá dirigir la petición, y el tercero es un acceso directo a la URL que se utiliza para acortar las URLS de petición en los *templates*, además de permitir una mejor gestión de las URLS. A continuación se describe la función en el archivo *views.py* a la cual se dirige la petición.

```
def login(request):
    if request.method == 'POST':
        context = RequestContext(request)
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                auth_login(request, user)
                return redirect(reverse('encuestas:home'))
            else:
                message = 'La cuenta de este usuario se encuentra deshabilitada'
                dict_response = {
                            'form': LoginForm(),
                            'message': message,
                            }
                return render_to_response('encuestas/login.html', dict_response, context)
        else:
            message = 'El usuario o el password son incorrectos'
            dict_response = {
                            'form': LoginForm(),
                            'message': message,
                            }
            return render_to_response('encuestas/login.html', dict_response, context)
    context = RequestContext(request)
    return render_to_response('encuestas/login.html', {'form': LoginForm(), }, context)
```

La función anterior obtiene los datos ingresados de un usuario y verifica que los mismos existan en la BD mediante una validación pre-establecida de Django "*authenticate*". Una vez validados los datos, se puede re-direccionar usando la disponibilidad del dispensador de direcciones, con la función "*redirect*" o bien indicando la dirección del *template* al cual se re-direccionará la petición, en este caso, con "*render_to response*".

4.2.4 Iteración: Generación de gráficas

Para la generación de gráficas se utilizó el paquete *python-xlsxwriter*. Contiene funciones para crear archivos *xlsx* de Excel y gráficas de la misma información ingresada [66]. El siguiente código describe la generación del archivo *xlsx* y las gráficas correspondientes a las tablas de datos.

```
def tabla_por_pregunta(pregunta, limite, colonia):
    # question data
    data = per_question(pregunta.pk, limite, colonia)

    worksheet_name = 'Pregunta %d' % (pregunta.pk)
    # Instance of workbook
    file_tittle = 'per_question_%d' % pregunta.pk
    _workbook = workbook_instance(worksheet_name, file_tittle)
    # ---> [_file, workbook, worksheet, worksheet_name, col, row]

    _file = _workbook[0]
    workbook = _workbook[1]
    worksheet = _workbook[2]

    #set BOLD style
    bold = workbook.add_format({'bold': True})

    _txt = u'Hoja de datos de la pregunta %s' % pregunta.pk
    worksheet.write(1, 2, _txt, bold)
    now = time.strftime("%c")
    _txt = 'Fecha: %s' % (now)
    worksheet.write(2, 2, _txt, bold)
    _txt = u'Límite: %s' % limite
    worksheet.write(3, 2, _txt, bold)
    _txt = u'Colonia: %s' % colonia
    worksheet.write(4, 2, _txt, bold)

    crear_xlsx_pregunta(pregunta, data, _workbook[1:])

    # Close document
    _workbook[1].close()

    _file = settings.XLSX_PATH_FILE + _workbook[0]
    return _file, _workbook[0]
```

Para generar el archivo *xlsx* se debe crear una instancia del mismo, asignando variables que indiquen el nombre del archivo *xlsx* (*file_tittle*) y el nombre de la hoja de cálculo (*worksheet_name*). Para poder agregar información al archivo se debe hacer enforma de coordenadas, tomando como primer valor la fila, el segundo la columna y el tercero el valor a ingresar, como valor opcional se puede agregar un cuarto parámetro que le da formato al texto ingresado, tal y como se indica en la siguiente instrucción:

```
_txt = u'Hoja de datos de la pregunta %s' % pregunta.pk
worksheet.write(1, 2, _txt, bold)
```

Para generar la gráfica se deben tener en cuenta los valores ingresados al archivo *xlsx* y sus posiciones, ya que el sistema de graficado trabaja en base a datos pre-insertados en el archivo y en base a ellos se realizan los cálculos.

```
chart1 = workbook.add_chart({'type': 'bar'})

# Configure the first series. Note use of alternative syntax to define
ranges.
chart1.add_series({
    'name':       [worksheet_name, 0, 2],
    'categories': [worksheet_name, in_datos - 1, col - 1 , fin_datos, col
- 1],
    'values':     [worksheet_name, in_datos - 1, col, fin_datos, col],
    })

# Add a chart title and some axis labels.
chart1.set_title ({'name': worksheet_name})
chart1.set_x_axis({'name': 'Encuestas'})
chart1.set_y_axis({'name': 'Manzana'})

# Set an Excel chart style.
chart1.set_style(11)

# Insert the chart into the worksheet (with an offset).
worksheet.insert_chart('D2', chart1, {'x_offset': 25, 'y_offset': 10})
```

El código anterior agrega una gráfica al archivo *xlsx* y después se le indican los parámetros de los datos ya existentes en el mismo archivo, para en base a ellos, generar el gráfico. El apartado "*name*" indica el nombre del gráfico a nivel archivo, "*categories*" indica las variables que tendrá la gráfica, en este caso *categories* serán las respuestas de cada pregunta, "*values*" serán el valor numérico de cada respuesta y que será representado en la gráfica.

Figura 4.13 Archivo *xlsx* generado por el sistema MiPYME-CAEO

Una vez ejecutado el código, se genera un archivo *xlsx*, que contiene los resultados de los valores ingresados, así como las gráficas correspondientes, tal y como se observa en la Figura 4.13.

4.2.5 Iteración: Generación de Directorio de Empresas

El directorio de empresas tiene como finalidad, el fácil acceso a la información del rubro de la empresa, tipo de giro, dirección, entre otros aspectos, esto es muy útil para identificar MiPYMES de algún giro en específico que ofrecen algún tipo de producto o servicio en el primer cuadrante de la ciudad de San Juan Bautista, Tuxtepec, Oaxaca.

La petición del CAEO para la generación del directorio fue poder filtrar las empresas por Tipo de Sector (Industrial, Comercial, Servicios) y tipo de empresa, es decir Micro, Pequeña o Mediana empresa, entre otros aspectos, permitiendo de esta manera facilitar la obtención de información en la BD. Los datos que se deben mostrar de la empresa son los siguientes: nombre, manzana, dirección, teléfono, sector, tipo de empresa, tipo de operación y tipo de negocio

Una vez obtenido el directorio de empresas, como se muestra en la Figura 4.14, se debe poder obtener un archivo de los datos de las empresas, esto mediante la descarga de tabla en formato *xlsx* de *Microsoft Office*, utilizando de igual manera el paquete *python-xlsxwriter*.

1	Directorio de Empresas							
2	Sector: Industrial							
3	Negocio: ---							
4	# Nombre	Manzana	Dirección	Teléfono	Sector	Tipo de Empresa	Tipo de Operació	Tipo de negocio
5	1 Caseta Telefónica e Internet	12	Eliseo Jimenez#99, La Piragua 68310		Industrial	Persona Física	Independiente	Micro
6	2 Abarrotes	109	23 de Septiembre#, María Luisa 68310		Industrial	Persona Física	Independiente	Micro
7	3 Tienda de Abarrotes	74	Mutualismo#83, Centro 68300		Industrial	Persona Física	Independiente	Micro
8	4 Plásticos Dany	13	20 de Noviembre#, La Piragua 68310		Industrial	Sociedad Anónima	Independiente	Micro
9	5 Estacionamiento	76	Libertad#0, Centro 68300		Industrial	Persona Física	Independiente	Micro
10	6 La Michoacana	21	Ocampo#, La Piragua 68310		Industrial	Persona Física	Sucursal	Micro
11	7 La Michoacana	44	Indepedencia#, Centro 68300		Industrial	Sociedad Anónima	Sucursal	Micro
12	8 Paletería la Flor de Puebla	64	Libertad#, Centro 68300		Industrial	Persona Física	Independiente	Micro
13	9 La Michoacana	69	5 de Mayo# , Centro 68300		Industrial	Persona Física	Franquicia	Micro
14	10 La Michoacana	69	Hidalgo# , Centro 68300		Industrial	Persona Física	Sucursal	Micro
15	11 Paletería la Michoacana	92	18 de Marzo#, Lázaro Cardenas 68313		Industrial	Persona Física	Independiente	Micro
16	12 La Flor del Trigo	31	Sebastian Ortiz#, La Piragua 68310		Industrial	Persona Física	Independiente	Micro

Figura 4.14. Directorio de empresas en archivo *xlsx*

4.3 Fase 3.- Aprendizaje

En esta fase se fueron programando reuniones parciales para ir presentando a los profesores del CAEO los avances en cada sección del cuestionario con el objetivo de identificar errores de captura, de llenado, de mala interpretación de las preguntas o de funcionalidad de los *templates* en el sistema MiPYME-CAEO, los cuales se fueron corrigiendo de acuerdo a las necesidades del cuestionario. Además se identificaron mejoras de diseño y funcionalidad del sistema las cuales se fueron realizando, probando y posteriormente se visualizó su funcionalidad en el sistema.

4.3.1 Cédula de identificación del negocio

Una de las primeras actualizaciones que se hizo en la cédula del negocio, fue la posibilidad de agregar más de un teléfono del negocio, ya que al principio solo se aceptaba uno. Esta fue una petición del CAEO, debido a que muchas empresas contaban con más de un teléfono de contacto. La siguiente actualización se hizo en la interfaz de usuario en el campo "colonia" al momento de llenar la cédula de la empresa. Esto agregando automáticamente el código postal de la colonia en la cual se encuentra el negocio.

Después de una de las reuniones realizadas con el grupo CAEO, se decidió modificar el modelo de los giros, esto debido a una actualización que realizaron a la tabla de giros, para efectos de tener los giros en una estructura más ordenada. El modelo "Giro" ahora tendría un segundo campo, llamado "tipo". En el siguiente código se presenta el modelo "Giro" con el nuevo campo.

```
class Giro(models.Model):
    giro = models.CharField(max_length=10)
    tipo = models.CharField(max_length=50)
    detalle = models.CharField(max_length=50)
```

Ahora con esta actualización se podrán filtrar los giros utilizando los dos primeros campos. Por ejemplo, se selecciona el giro "Industrial" y el tipo de giro "Alimentos y bebidas" y se obtendrán los giros específicos como se ejemplifica en la tabla de la Figura 4.15.

Giro	Tipo	Detalle
Industrial	Alimentos y bebidas	Fábrica de hielo
Industrial	Alimentos y bebidas	Panadería
Industrial	Alimentos y bebidas	Paletas, helados y aguas preparadas
Industrial	Alimentos y bebidas	Purificadora y embotelladora de agua
Industrial	Alimentos y bebidas	Tortillería

Figura 4.15 Tabla de representación de giros específicos de las MiPYMES

Algunas preguntas solo las puede contestar el propietario de la empresa, por lo que se solicitó agregar un filtro extra, que se aplica según la persona que esté respondiendo la encuesta, por ejemplo la pregunta 31, solo puede ser contestada si el propietario de la empresa está respondiendo la encuesta.

Por otra parte algunas de las empresas no estuvieron disponibles para contestar la encuesta, o simplemente no accedieron a contestarla, por lo que todas las preguntas quedaban vacías en ese caso específico. Esto representaba un espacio innecesario en la BD, por lo que se agregó la posibilidad de dejar una cédula de negocio cerrada. Esto es, que el encuestador pueda seleccionar la opción de cerrar la cédula de negocio y el sistema automáticamente almacena solo la cédula, sin las respuestas de las preguntas.

Además se solicitó que se generaran archivos en los cuales se obtuvieran los datos de las cédulas de las empresas y se contabilizaran. Para esto se deben obtener datos de las cédulas para todas las empresas y aplicando filtros por colonia, como se muestra en la Figura 4.16.

3	Encuesta perteneciente a la colonia: La Piragua	
4	Fecha: Mon Mar 9 16:59:42 2015	
5	Con un total de: 100 encuestas	
6	Encuestas abiertas: 100	
7	Encuestas cerradas: 0	
8		
9		
10	Tabla de Encuestas por Manzana	
11	Column1	
12	Manzana 2	6
13	Manzana 3	4
14	Manzana 5	1
15	Manzana 6	5
16	Manzana 7	1
17	Manzana 8	4
18	Manzana 9	6
19	Manzana 10	6
20	Manzana 12	1
21	Manzana 13	1
22	Manzana 14	4
23	Manzana 15	5
24	Manzana 16	4
25	Manzana 17	2

Figura 4.16 Documento *xlsx* con los datos de las cédulas filtradas para la colonia "La Piragua"

Los archivos son en formato *xlsx* y de igual manera se descargan mediante la interfaz de usuario directamente a una computadora.

4.3.2 Tipos de preguntas

Para que el sistema tuviera mejor fluidez y la BD fuera sencilla de gestionar, se decidió dividir las preguntas en 3 Tipos:

1. Tipo 1. Son todas esas preguntas en las que las respuestas solo se encuentran conformadas por una frase. Por ejemplo la pregunta 13 se muestra en la Figura 4.17, donde todas las posibles respuestas, están conformadas por una sola frase.

13.- ¿Qué día(s) de la semana cierra su establecimiento? (Puede señalar más de una opción)

Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo	Ningún día	Días festivos

Figura 4.17 Estructura de una pregunta tipo 1

2. Tipo 2. Son las preguntas cuyas respuestas están conformadas por dos frases. Por ejemplo la pregunta 25 se muestra en la Figura 4.18, donde se debe escoger entre Si y No, y después se seleccionan las respuestas correspondientes a la primera selección.

25.- ¿Considera adecuada la ubicación de su negocio y a qué factores lo atribuye? (Puede señalar más de una opción)

Si	Local céntrico	Cercanía de otros negocios	Cercanía de hogares	Transito constante de personas	Poca competencia	Seguridad	Otros:
No	Lejanía del local	Lejanía de otros negocios	Lejanía de hogares	Poco tránsito de personas	Mucha competencia	No hay seguridad	Otros:

Figura 4.18 Estructura de una pregunta tipo 2

3. Tipo 3. Son preguntas con respuestas compuestas por 3 filtros, por ejemplo en la pregunta 9, como se muestra en la Figura 4.19, se debe escoger una red social, después se debe indicar la identificación de la red social y posteriormente la frecuencia con la que se utilizan.

9.- Indique si el negocio tiene presencia en redes sociales y su frecuencia de uso. (Puede señalar más de una opción). En caso de ser "Ninguno" o "No tiene conocimiento", pasar a la pregunta 11.

Red Social	Indique el nombre de la identificación	Frecuencia				
		Diario	Semanal	Quincenal	Mensual	Otro:
Facebook						
Twitter						
YouTube						
Google+						
Otro:						
Ninguno:	( )	No tiene conocimiento ( )				

Figura 4.19 Estructura de una pregunta Tipo 3

Para poder generar gráficas de las preguntas con respuestas abiertas, se tuvo que poner como condición que se encasilláran en un solo tipo de respuesta, denominada "otros". Esto debido a que para muchas preguntas las gráficas se tornaban un poco ilegibles, por las diversas respuestas abiertas que se contabilizaban como una sola y que el sistema de graficación no podía descifrar con certeza, como por ejemplo la pregunta 11 que se muestra en la Figura 4.20.

11.-¿Cuáles son las razones por las cuales no tiene presencia el negocio en alguna red social? (Puede señalar más de una opción)

Desconoce la tecnología ( )	Tengo otras prioridades ( )	No incrementa mis ventas ( )	No hay personal dedicado a este fin ( )	No se requiere ( )	No tiene conocimiento ( )	Otro:

Figura 4.20 Estructura de una pregunta con respuesta abierta

La pregunta 11 tiene opción de respuesta abierta, pero al momento de graficar la pregunta se volvía poco legible, pero con la actualización se obtuvo una mejor gráfica y más accesible, como se muestra en la Figura 4.21.

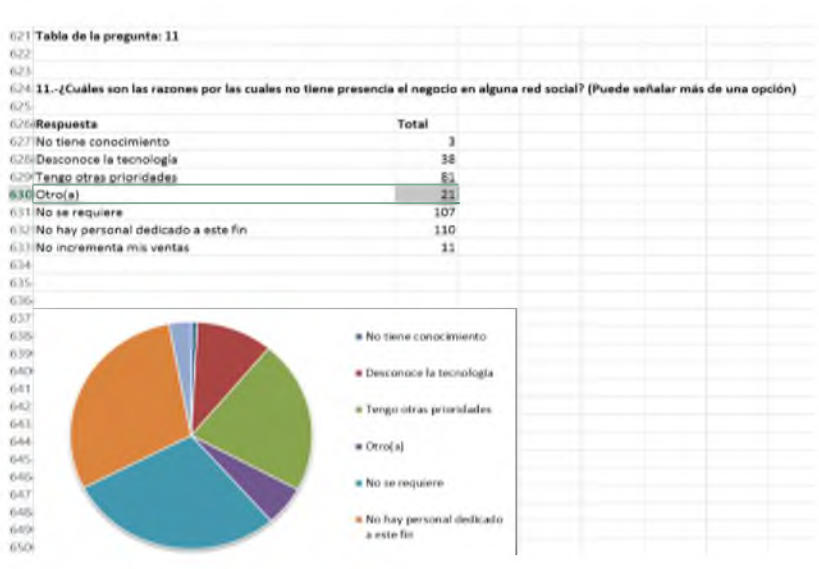


Figura 4.21 Archivo *xlsx* de una pregunta abierta

En las versiones nuevas de los archivos de gráficas, se ordenaron las respuestas conforme se muestran en las preguntas y el formato de la gráfica fue de tipo pastel. Para algunas preguntas se requiere de más de una gráfica, el CAEO solicitó que las gráficas fueran ordenadas conforme las respuestas, por ejemplo la pregunta 7. En esta pregunta se solicita al encuestado que indique si el negocio cuenta con página *Web* y también la frecuencia de uso.

Figura 4.22 Gráficas generadas de una pregunta con tres respuestas y frecuencia de uso

En ese caso la primera gráfica estaría conformada por las respuestas Si, No y No tiene conocimiento, tal y como se muestra en la Figura 4.22. La segunda gráfica estaría

conformada por las respuestas Diario, Semanal, Quincenal, Mensual, que corresponden directamente a contestar SI en la pregunta.

4.3.3 Directorio de Empresas

Después de entregar los archivos de prueba generados para el directorio de empresas, se decidió realizar lo siguiente:

- La columna de Manzana será completamente eliminada.
- La columna de Dirección se debe separar en Dirección, Colonia y CP.
- Con el cambio realizado a la tabla de Giros, ahora se deben agregar dos columnas más con los datos pertenecientes a la sección y el giro específico.
- El texto de la columna "Tipo de Empresa" ahora será "Tipo de Persona".
- El texto de la columna "Tipo de Negocio" ahora será "Tipo de Empresa".

Al realizar estos cambios como se presenta en la Figura 4.23, se obtiene un directorio mejor estructurado y con información más detallada de las empresas.

Directorio de Empresas - Sistema MiPYME-CAEO

Cuerpo Académico Estudios Organizacionaies - Universidad del Papaloapan

#	Nombre	Dirección	Colonia	CP	Teléfono	Sector	Sección	Giro Específico	Tipo de Persona	po de Operacic	Tipo de Empresa
1	Óptica Diamante	Libertad #408	Centro	68300	8753370	Comercial	Higiene y Salud	Óptica	Persona Física	Independiente	Micro
2	Antojitos Juquilita	20 de Noviembre #14	Centro	68300		Servicios	Alimentos y bebidas	Fonda y/o cocina económica	Persona Física	Independiente	Micro
3	Hotel Moctezuma	Privada Mootezuma #20	La Piragua	68310		Servicios	Turismo y actividades relacionadas	Hotel	Persona Física	Independiente	Micro
4	Farmacia Albatros	Carranza #1554	La Piragua	68310		Comercial	Higiene y Salud	Farmacia	Persona Física	Sucursal	Micro
5	Renovadora de Calzado	Indepencencia #1687	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Renovadora de calzado	Persona Física	Independiente	Micro
6	Farmapronto	Carranza #1573	La Piragua	68310		Comercial	Higiene y Salud	Farmacia	Persona Física	Sucursal	Micro
7	Hojalatería Pulido Trujillo	Indepencencia #1753	La Piragua	68310		Industrial	Bienes de consumo duradero	Hojalatería	Persona Física	Independiente	Micro
8	Más que un Regalo	Indepencencia #1883	La Piragua	68310		Comercial	Bienes de lujo y esparcimiento	Tiendas de regalos	Persona Física	Independiente	Micro
9	Financiera FINSOL	Indepencencia #1655	La Piragua	68310	1063650	Servicios	Administrativos, financieros y legales	Entidades financieras y seguros	Sociedad Cooperativa	Sucursal	Micro
10	Frenos y Clutch Yucos	Zaragoza #711	La Piragua	68310	287 104 54 58	Servicios	Servicios de mantenimiento y reparación	Taller automotriz mecánico y eléctrico	Persona Física	Independiente	Micro
11	Global Comunicaciones	Libertad #1925	La Piragua	68310		Servicios	Telecomunicaciones y transportación	Telefonía y comunicaciones	Persona Física	Sucursal	Micro
12	Micronegocio Azteca	Libertad S/N	La Piragua	68310	87 5 84 30	Servicios	Administrativos, financieros y legales	Entidades financieras y seguros	Sociedad Anónima	Sucursal	Micro
13	Refaccionaria los Tres Pistones	Rivapalacio S/N	La Piragua	68310	287 888 42 20	Comercial	Productos eléctricos y de electrónica	Electrónica	Persona Física	Independiente	Micro
14	Rectificadora Oriente	Rivapalacio #1928	La Piragua	68310	87 5 09 84	Servicios	Servicios de mantenimiento y reparación	Taller de artes plásticas	Persona Física	Independiente	Micro
15	Maderas Papaloapan	Reforma #603	La Piragua	68310	8754848	Industrial	Bienes de consumo duradero	Carpintería	Persona Física	Independiente	Micro
16	Clínica Fentanez	Reforma S/N	La Piragua	68310		Servicios	Higiene y Salud	Hospital	Persona Física	Independiente	Pequeña
17	Despacho de Abogados	Independencia S/N	La Piragua	68310		Servicios	Administrativos, financieros y legales	Abogado	Persona Física	Independiente	Micro
18	Bar la Abuelita	Arista S/N	La Piragua	68310		Servicios	Alimentos y bebidas	Bar, cantina y similar	Persona Física	Independiente	Micro
19	Lavado de Autos y Aspirado "La Piragua"	Reforma S/N	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Lavado de autos	Persona Física	Independiente	Micro
20	Taller de Tornos y Soldadura	Independencia S/N	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Taller automotriz mecánico y eléctrico	Persona Física	Independiente	Micro
21	Laminación y Pintura	Independencia #1419	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Taller de laminación y pintura	Persona Física	Independiente	Micro
22	Consultorio del Dr. Patatuchi	Independencia #1753	La Piragua	68310		Servicios	Higiene y Salud	Consultorio para el cuidado de la salud	Persona Física	Independiente	Micro
23	Marisquería Rulos	Independencia S/N	La Piragua	68310		Servicios	Alimentos y bebidas	Restaurant	Persona Física	Independiente	Micro
24	Estafeta	Independencia #1502	La Piragua	68310		Servicios	Telecomunicaciones y transportación	Servicio postal, mensajería y paquetería	Sociedad Anónima	Sucursal	Micro
25	Taller de Motos	Independencia S/N	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Taller automotriz mecánico y eléctrico	Persona Física	Independiente	Micro
26	Refaccionaria González	Rivapalacios #10678	La Piragua	68310		Comercial	Bienes de consumo duradero	Accesorios y/o refaociones automotrices	Persona Física	Independiente	Micro
27	Taquería el Tapatio	20 de Noviembre #1705	La Piragua	68310	8754059	Servicios	Alimentos y bebidas	Taquería	Persona Física	Independiente	Micro
28	Tlapalería El Fierrito	20 de Noviembre #1612	La Piragua	68310	8753290	Comercial	Bienes de consumo duradero	Ferretería y tlapalería	Persona Física	Independiente	Micro
29	Clínica Fentanez (Farmacia)	Reforma #493	La Piragua	68310	8757748	Comercial	Higiene y Salud	Farmacia	Persona Física	Independiente	Micro
30	Cheveria Independencia	Independencia S/N	La Piragua	68310		Comercial	Alimentos y bebidas	Depósito de cerveza	Persona Física	Independiente	Micro
31	Consultorio Oftalmológico	Independencia #1513	La Piragua	68310		Servicios	Higiene y Salud	Consultorio para el cuidado de la salud	Persona Física	Independiente	Micro
32	Carmelita	Independencia #309	La Piragua	68310		Servicios	Alimentos y bebidas	Fonda y/o cocina económica	Persona Física	Independiente	Micro
33	Casa Mazatlán	Independencia #725	La Piragua	68310		Servicios	Administrativos, financieros y legales	Entidades financieras y seguros	Sociedad Anónima	Sucursal	Micro
34	Infantiles Ilusión	Independencia #691	La Piragua	68310		Comercial	Artículos de uso personal	Ropa nueva	Persona Física	Independiente	Micro
35	Depósito Cheva	Reforma S/N	La Piragua	68310		Comercial	Alimentos y bebidas	Depósito de cerveza	Persona Física	Independiente	Micro
36	Taller de Soldadura S/N	Reforma S/N	La Piragua	68310		Servicios	Servicios de mantenimiento y reparación	Taller automotriz mecánico y eléctrico	Persona Física	Independiente	Micro
37	Super Bella	Independencia #665	La Piragua	68310		Comercial	Artículos de uso personal	Ropa nueva	Persona Física	Independiente	Micro
38	Sabrosísimo	Reforma S/N	La Piragua	68310		Servicios	Alimentos y bebidas	Restaurant	Persona Física	Independiente	Micro

Figura 4.23 Estructura del directorio de empresas

Capítulo 5. Conclusiones

5.1 Conclusiones generales

Con el sistema MiPYME-CAEO se obtienen datos relevantes sobre los negocios que actualmente existen en el primer cuadrante de la ciudad de San Juan Bautista Tuxtepec, Oaxaca, tales como los tipos de empresas que predominan en la ciudad y el nivel de escolaridad que tienen los empleados. También se genera un directorio de empresas que facilita la localización y clasificación de las empresas, contribuyendo a la toma de decisiones empresariales.

La metodología de desarrollo de software ASD fue muy importante en este proyecto, ya que gracias a la interacción constante entre el desarrollador y el equipo del CAEO, se generó un sistema perfectamente acoplado a las necesidades del usuario final, robusto y que proporcionó los resultados esperados.

Django ha probado ser una herramienta que agiliza el desarrollo de aplicaciones *Web* ahorrando mucho tiempo de programación, además de promover el uso claro y comprensible del código mediante funciones predefinidas que son primordiales para una aplicación *Web*, conjuntamente con el estilo de programación de *Python*.

Por otra parte *Python* es uno de los lenguajes de programación más sencillos de aprender y uno de los más robustos al mismo tiempo, por lo que no se requiere de vasta experiencia para poder lidiar con él y se tiene la seguridad de obtener un sistema de alta calidad.

En relación al gestor de bases de datos, MySQL sobresale entre los diversos SGBD como uno de los más utilizados, sin embargo, no se requiere tener gran conocimiento sobre MySQL ya que cualquier aplicación ejecuta *querys* utilizando los modelos de *Django* y es completamente funcional, lo cual permite que usuarios sin conocimientos en SGBD utilicen la herramienta fluidamente. *Django* no requiere de configuraciones extras para desplegarse en producción, a excepción de la conexión con

un servidor, en este caso Apache, que fue utilizado para ejecutar la aplicación en un servidor local.

5.2 Contribuciones de la tesis

Este trabajo tiene como principal contribución la generación de un directorio de las MiPYMES del primer cuadrante de la ciudad de Tuxtepec, Oaxaca generado con el sistema MiPYME-CAEO desarrollado con el Framework Django. Sin embargo también se obtuvieron los siguientes resultados:

1. Análisis, diseño y elaboración de la estructura de la BD.
2. Adaptación de la estructura de la BD conforme a los modelos del *Framework* Django.
3. Elaboración de los t*emplates* en base a un instrumento de recolección de datos para MiPYMES de la región de la cuenca del Papaloapan.
4. Generación de documentos *XLSX* para el análisis de resultados por parte de los integrantes del CAEO, de acuerdo al área de tecnologías de la información, mercadotecnia, administración y finanzas de las MiPYMES.

5.3 Trabajo a futuro

Como trabajo a futuro, se planea desplegar el sistema en un servidor *Web,* al cual se pueda tener acceso desde cualquier parte de la ciudad de San Juan Bautista Tuxtepec, Oaxaca. El mismo sistema no requiere de mayor actualización, salvo agregar nuevas preguntas y/o modificar las ya existentes, por lo que el CAEO lo piensa utilizar para obtener los datos de los sectores restantes y con eso complementar el directorio de las empresas de toda la ciudad de San Juan Bautista Tuxtepec, Oaxaca.

Al ser accesado el sistema vía internet, se pretende crear un apartado libre para el público en general, de esta manera cualquier persona podra acceder al directorio de empresas, al cual se le pretenden agregar características que faciliten la localización de empresas por medio de rubros, ubicaciones, años de servicio, entre otros.

Aprovechando las nuevas tecnologías de geolocalización, los usuarios que busquen una empresa podrán ubicarla por medio de su dirección y con la ayuda de

sistemas como *Google Maps*. De esta manera el usuario tendrá la ubicación exacta del negocio.

Acrónimos y Términos usados

MiPYME. Micro, Pequeñas y Medianas Empresas.

CAEO. Cuerpo Académico Estudios Organizacionales.

Framework. Conjunto de herramientas, prácticas y definiciones enfocadas a la resolución de un problema en particular y similares.

Django. *Framework* de desarrollo *Web*, que utiliza *Python* como lenguaje de programación base.

Template. Interfaz utilizada por *Django* para acoplar el uso de su lenguaje con etiquetas HTML.

ORM. *Object Relational Mapping*. Mapeado Objeto-Relacional.

AJAX. *Asynchronous JavaScript And XML*. Método de programación *Web*, donde se realizan peticiones a servidor en Segundo plano.

MVC. Modelo Vista Controlador

MTV. Model Template View.

CSV. *Comma Separated Values*, formato de archivo que almacena los datos a manera de tablas.

Python. Lenguaje de programación de alto nivel.

BD. Base de Datos, las cuales son colecciones de información alojada en estructuras llamadas tablas.

SGBD. Sistema Gestor de Bases de Datos, los cuales se encargan de gestionar los datos alojados en bases de datos.

ER. Modelo Entidad-Relación.

EER. Modelo Entidad-Relación Mejorado.

HTML. *Hyper Text Markup Languaje*. Lenguaje de marcado de hipertexto

XLSX. Extensión de archivo con formato XML para hojas de cálculo utilizado por *Microsoft Excel*.

Javascript (JS). Lenguaje de programación interpretado.

JQuery. Librería de funciones JS.

POST. Método de traspaso de información a determinada función.

WWW. *World Wide Web*, sistema de búsqueda por medio de Internet.

URL. *Uniform Resource Locator*, que representa la dirección de recursos en la WWW.

UTF-8. *8-bit Unicode Transformation Format*. Formato de codificación de caracteres Unicode.

Xslxwriter. Módulo de *Python* para el desarrollo de archivos XLSX.

CIN. Cédula de Identificación de Negocio.

Query. Pregunta o consulta estructurada para la obtención de información.

SQL. *Structured Query Languaje*, Lenguaje de Consulta Estructurado.

MySQL. Es un SGBD relacional, multihilo y multiusuario con más de seis millones de instalaciones

SQLite. Es un SGBD relacional compatible con ACID, contenida en una pequeña biblioteca escrita en C.

Apache. Servidor *Web* HTTP de código abierto

WSGI. *Web Server Gateway Interface*, es una interfaz simple y universal entre los servidores *Web* y aplicaciones *Web* o marcos para el lenguaje de programación *Python*.

Referencias

1. Pro México Inversión y Comercio. http://www.promexico.gob.mx/negocios-internacionales/pymes-eslabon-fundamental-para-el-crecimiento-en-mexico.html. Consulta en línea [28/04/2015].
2. H. Ayuntamiento Constitucional de San Juan Bautista Tuxtepec, Oaxaca. http://www.tuxtepec.gob.mx. Consulta en línea [28/04/2015].
3. Why We Choose Python. http://www.sixfeetup.com/blog/why-we-choose-python. Consulta en línea [03/05/2015].
4. Tango with Django 1.7. http://www.tangowithdjango.com/. Consulta en línea [03/05/2015].
5. Top Reasons to Use MySQL. https://www.mysql.com/why-mysql/topreasons.html. Consulta en línea [03/05/2015].
6. José Luis Cendejas Valdéz, Carlos Arturo Vega Lebrún, Anayansi Careta Isordia, Osvaldo Gutiérrez Sánchez, Heberto Ferreiro Medina. "Diseño del modelo integral colaborativo para el desarrollo ágil de software en las empresas de la zona centro-occidente en México". Revista Electrónica Nova Scientia, Vol. 7, Núm. 13 (2014).
7. Perozo Arnoldo José, Villalobos Rixia. "Naturaleza de la innovación tecnológica en las organizaciones de desarrollo de software como servicio del municipio Maracaibo". Universidad Privada Dr. Rafael Belloso Chacín, Vol. 13, Núm. 2 julio – diciembre (2014).
8. Vandivier Steve y Cox Kelly. "Manual de Oracle 9. Aplication Server Portal". Mc Graw-Hill/Interamericana de España, S.A.U. ISBN: 84-481-3253-X, 2002
9. Tejada Artigas Carlos, Rodríguez Yunta Luis. "Empresas españolas de servicios documentales: clasificación, tipología de servicios y encuesta sobre empleo". El profesional de la información, Vol. 13, Núm. 6. noviembre - diciembre (2004).
10. Gálvez Albarracín Edgar Julián, Riascos Erazo Sandra Cristina, Contreras Palacios Fred. "Influencia de las tecnologías de la información y comunicación en el rendimiento de las micro, pequeñas, y medianas empresas colombianas". Universidad ICESI. Elsevier España, S.L.U. (2013).

11. FarmaIndustria. “Directorio de empresas españolas de recientecreación que desarrollan su actividad en el área de la biotecnología aplicada a la salud humana”. Medicamentos Innovadores, Plataforma Tecnológica Española. Junio (2010).
12. Ibarra Morales Luis Enrique, Casas Medina Emma Vanessa, Olivas Valdez Erika, Espinoza Galindo Belén. “El comercio electrónico en las pequeñas y medianas empresas comerciales (pymes) que operan en Hermosillo, Sonora”. Global Conference on Business and Finance Proceedings, Vol. 9, Núm. 2. (2014).
13. López Ponce María Eugenia, Suárez Améndola Rosario de Fátima. “Bases de Datos en las pequeñas y medianas empresas (pymes)”. Fomix Campeche Revista, pp. 43-44. Abril – Junio (2012).
14. Velarde López Elvira, Araiza Garza Zóchitl, Hernández Castro Nidia, Tobías Sierra Lluvia. “Estrategias de dirección y tecnologías de información en pequeñas y medianas empresas de Coahuila”. Revista internacional administración & finanzas, Volumen 4, Numero 1, (2011).
15. Ihme Tuomas, Pikkarainen Minna,Teppola Susanna, Kääriäinen Jukka, Biot Olivier. “Challenges and industry practices for managing software variability in small and medium sized enterprises”. Springer Science + Business Media. New York (2013).
16. Seethamraju Ravi. “Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs)”. Springer Science+Business Media. New York (2014).
17. Figueroa González Ernesto Geovani, Hernández Cantú Flor Isela, González Herrera María Brenda, Arrieta Díaz Delia. “Comercio electrónico como factor competitivo en las micro, pequeñas y medianas empresas del sector comercial en el estado de Durango”. Revista internacional administración & finanzas, Volumen 6, Numero 3, (2013).
18. Bath Timothy G., Bozdag Selcuk, Afzal Vackar y Crothers Daniel. “LimsPortal and BonsaiLIMS: development of a lab information management system for translational medicine”. Source Code for Biology and Medicine. Pp 1-5, ISSN 1751-0473. BioMed Central (2011).

19. Khalid Khan Mohammed, Sohail Muhammad, Aamir Muhammad, Chowdhry B.S., Irfan Hyder Syed. "Web Support System for Business Intelligence in Small and Medium Enterprises". Wireless Personal Communications, Volumen 75, Issue 3, pp. 535-548, ISSN 0929-6212. Springer US (2014).
20. Elina Mendoza Martha, Durán Meneses Lorena y Rivera Ortíz Norma. "Metodología de Desarrollo de Bodegas de Datos Para Micro, Pequeñas y Medianas Empresas". UIS Ingenierías, Volumen 9, Número 1, pp. 85-101. Facultad de Ingenierías Físico Mecánicas (2010).
21. Code Project. "What Is A Framework", http://www.codeproject.com/Articles/5381/What-Is-A-Framework. Consulta en línea [15/05/2015]
22. What Is "Framework", http://whatis.techtarget.com/definition/framework. Consulta en línea [15/05/2015]
23. Docs Springs. "Spring Framework Reference Documentation". http://docs.spring.io/spring/docs/current/spring-framework-reference/html/index.html. Consulta en línea [20/05/2015]
24. IBM Knowledge Center. "Objetos POJO (Plain old Java objects)", http://www-01.ibm.com/support/knowledgecenter/SS4JE2_7.5.5/org.eclipse.jst.ejb.doc.user/topics/cpojosandee5.html?lang=es. Consulta en línea [21/05/2015]
25. On Java. "Introduction to Aspect-Oriented Programming", http://www.onjava.com/pub/a/onjava/2004/01/14/aop.html.Consulta en línea [22/05/2015]
26. Zend Framework. "Zend Framework Quick Start", http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html. Consulta en línea [22/05/2015]
27. C2. "Modelo View Controller", http://c2.com/cgi/wiki?ModelViewController. Consulta en línea [22/05/2015]
28. CODEHERO. "PHP desde cero: PHP Unit I", http://codehero.co/php-desde-cero-phpunit-i/. Consulta en línea [22/05/2015]
29. Tuts+. "Travis-CI: What, Why, How", http://code.tutsplus.com/tutorials/travis-ci-what-why-how--net-34771. Consulta en línea [22/05/2015]

30. Rails Guides. "Getting Started with Rails", http://guides.rubyonrails.org/getting_started.html. Consulta en línea [25/05/2015]
31. Ruby. "Libraries", https://www.ruby-lang.org/en/libraries/. Consulta en línea [25/05/2015]
32. Ruby Gems. "Stats", https://rubygems.org/stats. Consulta en línea [25/05/2015]
33. Ruby on Rails. "Deploy Ruby on Rails is easy", http://rubyonrails.org/deploy/. Consulta en línea [25/05/2015]
34. Michael Hartl. "Ruby on Rails Tutorial (3rd Edition): Learn Web Development with Rails", https://www.railstutorial.org/. Consulta en línea [25/05/2015]
35. Ruby. "About Ruby", https://www.ruby-lang.org/en/about/. Consulta en línea [25/05/2015]
36. Django Project. "Why Django?", https://www.djangoproject.com/start/overview/. Consulta en línea [10/06/2015]
37. Python Software Foundation. "Functional Programming HOWTO", https://docs.python.org/2/howto/functional.html. Consulta en línea [10/06/2015]
38. Django Book. "Chapter 1.Introductionto Django", http://www.djangobook.com/en/2.0/chapter01.html. Consulta en línea [10/06/2015]
39. The Django Book."Chapter 5: Models", http://www.djangobook.com/en/2.0/chapter05.html#the-mtv-or-mvc-development-pattern. Consulta en línea [10/06/2015]
40. Silberschatz, A, Korth, H.F, Sudarshan, "Fundamentos de bases de datos (3ra ed.)". McGraw-Hill 1998.
41. Mark Mcllroy, "SQL essentials". Blue Sky Technology 2009.
42. James R. Groff, Paul N Weinberg, Andrew J. Oppel, "SQL, The Complete Reference 3rd Edition". McGrawGill 2010.
43. Oracle. "Introduction to the Oracle Database", http://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm. Consulta en Linea [14/06/2015].
44. Techopedia. "Oracle Database (Oracle DB)", http://www.techopedia.com/definition/8711/oracle-database. Consulta en línea [14/06/2015]

45. Microsoft.“Microsoft SQL Server”, https://msdn.microsoft.com/en-us/library/bb545450.aspx. Consulta en línea [15/06/2015]
46. PostgreSQL. “PostgreSQL 9.4.4 Documentation”, http://www.postgresql.org/docs/9.4/static/index.html. Consulta en línea [15/06/2015]
47. MySQL. “Chapter 1 General Information”, http://dev.mysql.com/doc/refman/5.7/en/introduction.html. Consulta en línea [15/06/2015]
48. MySQL. “TheMyISAM Storage Engine”, http://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html. Consulta en línea [18/06/2015]
49. MySQL. “IntroductiontoInnoDB”, http://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html. Consulta en línea [18/06/2015]
50. Mozilla. “Whatis a web server”, https://developer.mozilla.org/en-US/Learn/What_is_a_web_server. Consulta en línea [25/06/2015]
51. Microsoft. “Introductionto IIS Architectures”, http://www.iis.net/learn/get-started/introduction-to-iis/introduction-to-iis-architecture. Consulta en línea [25/06/2015]
52. NGINX. “Beginner´s Guide”, http://nginx.org/en/docs/beginners_guide.html. Consulta en línea [25/06/2015]
53. LITESPEED. “Documentation”, https://www.litespeedtech.com/docs/webserver/introduction. Consulta en línea [25/06/2015]
54. Apache.“HTTP SERVER PROJECT”, http://httpd.apache.org/. Consulta en línea [25/06/2015]
55. Wiki Apache.“What Is Apache”, http://wiki.apache.org/httpd/FAQ#What_is_Apache.3F. Consulta en línea [20/06/2015]
56. Szalvay Victor. “An Introduction to Agile Software Development”. http://www.danube.com/docs/Intro_to_Agile.pdf. Consulta en línea [20/09/2015]

57. Ticona Condori Shirley Fabiola. "METODOLOGÍAS TRADICIONALES, METODOLOGÍAS ÁGILES, METODOLOGÍAS PARA JUEGOS, METODOLOGÍAS EDUCATIVAS Y METODOLOGÍAS PARA APLICACIONES MÓVILES". https://tallerinf281.wikispaces.com/file/view/METODOLOG%C3%8DAS+TRADICIONALES.pdf. Consulta en Línea [18/05/201]
58. Trigas Gallego Manuel. "Metodología SCRUM". http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf. Consulta en Línea [21/09/2015]
59. Palacio Juan."El Modelo SCRUM". http://www.navegapolis.net/files/s/NST-010_01.pdf1. Consulta en Línea [21/09/2015]
60. Voigt J. J. Benjamin. "Dynamic System Development Method". https://files.ifi.uzh.ch/rerg/arvo/courses/seminar_ws03/14_Voigt_DSMD_Ausarbeitung.pdf. Consulta en Línea [24/09/2015]
61. Ramsin Raman. "Software Development Methodologies, Lecture 8. Agile Methodologies: DSDM". http://sharif.edu/~ramsin/index_files/sdmlecture8.pdf. Consulta en línea [25/09/2015]
62. Acebal Cesar F., Cueva Lovelle Juan M. "Extreme Programming (XP): un nuevo método de desarrollo de software". Depto. de Informática, Área de Lenguajes y Sistemas Informáticos, Universidad de Oviedo. NOVATICA, Marzo Abril 2002, pp 8-12
63. Joskowicz José. "Reglas y Prácticas en eXtremeProgramming". http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf. Consulta en línea [02/10/2015]
64. Keith Everette R. "Agile Software Development Processes A Different Approach to Software Design". https://www.cs.nyu.edu/courses/spring03/V22.0474-001/lectures/agile/AgileDevelopmentDifferentApproach.pdf. Consulta en línea [10/10/2015]
65. Ramsin Raman. "Software Development Methodologies, Lecture 11. Agile Methodologies: ASD". http://sharif.edu/~ramsin/index_files/sdmlecture11.pdf. Consulta en línea [29/09/2015]

66. Python. “XlsxWriter 0.8.4”.https://pypi.python.org/pypi/XlsxWriter. Consulta en línea [17/07/2015].