UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Ingeniería de Software y Sistemas Informáticos

Aplicación y evaluación de la metodología desarrollo orientado por pruebas (TDD), caso de estudio: spot.co.

Trabajo Fin de Máster

Presentado por: Torres Corredor, Oscar Iván

Director/a: Núñez Valdez, Edward Rolando

Ciudad: Duitama
Fecha: 2017

Resumen

La metodología de desarrollo de software orientado por pruebas (TDD) está tomando fuerza en el bosquejo y desarrollo de sistemas; su facilidad de implementación y calidad del producto lo hace un modelo excelente de adoptar por las empresas. En este trabajo se propone el desarrollo de una plataforma de anuncios publicitarios, creada bajo la metodología TDD, evaluando la calidad del producto, teniendo en cuenta las características del modelo de calidad del estándar de la ISO/IEC 9126, para determinar el nivel de aptitud de la plataforma, y dar constancia de que dicha metodología puede ser adoptada en el desarrollo de software por las empresas. Para el adelanto de la investigación se manejaron los marcos de trabajo XP (programación extrema), TDD (desarrollo orientado por pruebas) y el estándar de la ISO/IEC 9126. La implementación del modelo de calidad al software de anuncios publicitarios (spot) muestra un resultado prometedor en cuanto a las características evaluadas, sin embargo al finalizar la investigación se encontró que las pruebas de la metodología TDD solo funcionan en la capa de negocio, más no en la interfaz de usuario del software.

Palabras Clave: TDD, ISO/IEC9126, Calidad, Software, Modelo, Metodología.

Abstract

The methodology of development of software oriented by tests (TDD) is taking force in the design and development of systems; its ease of implementation and product quality makes it an excellent model to adopt by companies. In this work we propose the development of an advertising platform, created under the TDD methodology, evaluating the quality of the product, taking into account the characteristics of the quality model of the ISO / IEC 9126 standard, to determine the level of aptitude of the platform, and give proof that said methodology can be adopted in the software development by the companies. For the development of the research, the frameworks XP (extreme programming), TDD (development oriented by tests) and the standard of ISO / IEC 9126 were used. The implementation of the model of quality to the software of advertisements (spot) shows a promising result in terms of the characteristics evaluated, however at the end of the investigation it was found that the tests of the TDD methodology only work in the business layer, but not in the user interface of the software.

Keywords: TDD, ISO / IEC9126, Quality, Software, Model, Methodology

Índice de contenidos

Índice de tablas

Índice de figuras

Introducción

La aparición de los modelos, diseños, notaciones, herramientas y marcos de trabajo fueron aspectos importantes para el éxito en el desarrollo de software de una organización; sin embargo, la implementación de dichas herramientas a los procesos de un flujo de trabajo en la compañía, implicaba un proceso amplio de adaptación y normalización para alcanzar el nivel óptimo en el desarrollo de un sistema; dichos modelos, notaciones, marcos de trabajo y herramientas no cumplieron en su totalidad con la calidad de un producto de software y los requisitos que exigía el mercado en su momento. [1] Por lo tanto, se tuvo que replantear las metodologías tradicionales, por un nuevo modelo que permitiera solventar los requisitos cambiantes en el ciclo de vida de la creación de un software y la particularidad del producto; de tal forma y teniendo en cuenta las anteriores condiciones, emergen las metodologías agiles para dar solución a dichos inconvenientes y revolucionar la ingeniería del software.

Las metodologías agiles son actualmente el tema de moda en lo que respecta a la ingeniería del software; desde que se creó el manifiesto ágil en el año 2001 [2] han surgido diferentes metodologías, que permiten facilitar el desarrollo de software, teniendo en cuenta los entornos cambiantes de las necesidades de los clientes y la calidad del producto. Sin embargo, son tantos los marcos de trabajo que se han creado, que las diferentes empresas solo han podido implementar algunos de ellos; teniendo dentro de los más utilizados, las metodologías de Programación Extrema, Scrum, Kanban y Agile Inception [3] [4], pero no se ha analizado profundamente los beneficios que puede brindar las demás metodologías agiles, para ser implementadas en el desarrollo de sistemas informáticos. La presente investigación se realizó con el fin de profundizar en el marco de trabajo de desarrollo orientado por pruebas (TDD) y la evaluación de calidad del producto bajo las características del estándar de la ISO/IEC 9126, para determinar el nivel de confianza del software desarrollado, bajo dicha metodología. Como conclusión, el marco de trabajo TDD, mejora la calidad del producto, pero compromete la productividad del equipo desarrollador, debido a la cantidad de pruebas que se deben realizar como parte del proceso algorítmico, que propone dicha metodología. Sin embargo, la implementación del modelo de calidad de la ISO/IEC 9126 al producto, muestra un nivel de calidad óptimo.

El desarrollo de la investigación se realizó en tres espacios, el primer capítulo consistió en la caracterización del negocio y la identificación de las variables a sistematizar como parte de la plataforma de anuncios publicitarios (spot); en el segundo capítulo se implementó la metodología de desarrollo de software orientado por pruebas a la plataforma de anuncios

publicitarios, y finalmente en el tercer capítulo se diseñó e implementó el modelo de calidad de software, teniendo en cuenta las características de evaluación del estándar de la ISO/IEC 9126, para determinar la confianza que debe tener dicho marco de trabajo (TDD) para ser adoptado por las empresas que desarrollan software.

1. Generalidades

Para el desarrollo de la presente investigación es conveniente sentar las bases de la investigación, partiendo desde el planteamiento del problema, que conlleva a determinar el por qué se realizó la investigación y cuál es el aporte a la comunidad científica; luego se describe la justificación, las razones por las que es importante desarrollar el proyecto, y por último se describe de forma general la estructura del proyecto.

1.1. Planteamiento del problema

La evolución de las metodologías del ciclo de vida de desarrollo de software en las últimas décadas, han permitido desarrollar aplicaciones de calidad y en el menor tiempo posible. Actualmente las metodologías conocidas como agiles, que se crearon para ser empleadas como modelos óptimos en el desarrollo de software, están siendo adoptadas sigilosamente por las empresas, para desarrollar aplicaciones que sean acorde a los requisitos de lo que desea el cliente.

Con el paso del tiempo se han ido creando y perfeccionando un sin número de metodologías agiles para el desarrollo de software, sin embargo se tienen dentro de las más usadas, los procesos basados en SCRUM, Programación Extrema y Lean Kanban, sin llegar a profundizar en otros marcos de trabajo, que tal vez pueden ser eficientes en el proceso de desarrollo de software; por tal motivo se hace necesario desarrollar una aplicación web de anuncios publicitarios, basada en el marco de trabajo TDD (Desarrollo de software orientado por pruebas) y evaluar la particularidad del producto con las características del modelo de calidad de la IOS/IEC 9126 para determinar el nivel de confianza del producto y poder entrar a nuevos modelos de desarrollo de software, pero especialmente para ser adoptados por las empresas que crean aplicaciones.

1.2. Justificación

Los modelos agiles surgieron como medida alternativa, a las falencias que presentaban las metodologías tradicionales en su momento, especialmente en el desarrollo de software; los

procesos basados en el manifiesto ágil, se centraron fundamentalmente en permitir cambios, a los requisitos del sistema y así refinar la idea de negocio a través de iteraciones, para obtener un producto de calidad, al final del proceso. El modelo ágil fue y actualmente es un concepto revolucionario que ha adquirido fuerza y ha permitido desarrollar diferentes metodologías en el sistema ágil de software. Sin embargo, el éxito y aumento que han tenido dichas metodologías a nivel mundial; la acogida e implementación de dichos procesos, no han tenido el suficiente impacto; de las diferentes metodologías desarrolladas, se tienen algunas dentro de las más usadas, pero no se ha profundizado en otras metodologías que pueden ser eficientes al momento de desarrollar un software, por lo tanto es conveniente realizar un estudio para consolidar un marco de trabajo, nada común en el desarrollo de software en la actualidad por parte de las empresas.

El presente proyecto pretende determinar la calidad del producto de una aplicación web de anuncios publicitarios, creada utilizando la metodología ágil de desarrollo de software orientado por pruebas (TDD), teniendo en cuenta los criterios de evaluación del modelo de calidad, del estándar de la ISO/IEC 9126. El desarrollo del presente análisis, permite mostrar la importancia de escribir códigos de prueba de software, para obtener un código robusto, seguro, mantenible, y aumentar la rapidez de desarrollo, pero sobre todo, que se puede obtener un producto de calidad. El desarrollo del presente proyecto da a conocer la importancia, que tiene la metodología ágil de desarrollo orientado por pruebas (TDD), en la creación de productos de calidad, teniendo en cuenta, que también permite abrir las puertas para ser implementado en empresas de desarrollo de software, debido a que actualmente no es una metodología común, para ser usada por dichas entidades.

1.3. Estructura del proyecto

El proyecto de investigación se desarrolla, partiendo desde la descripción de los conceptos, teorías e investigaciones que se tienen cuenta para plasmar la idea del proyecto, luego se describe detalladamente las características que se deben tener en cuenta para implementar la metodología TDD a un proyecto de anuncios publicitarios y el proceso de evaluación de la plataforma de acuerdo al modelo de calidad del estándar ISO/IEC 9126. Seguidamente, se documenta el desarrollo de cada uno de los objetivos del proyecto, el proceso de la implementación de la metodología ágil TDD al desarrollo de la plataforma de anuncios publicitarios, luego se documenta el diseño de la guía de pruebas, para evaluar el sistema de acuerdo a las métricas del modelo de calidad del estándar de la ISO/IEC 9126. Finalmente se documentan las conclusiones y las líneas de trabajo para proyectos futuros.

2. Marco referencial

Este capítulo se documenta las investigaciones que se tuvieron en cuenta, para el desarrollo del presente proyecto de grado. Inicialmente se describen las diferentes investigaciones que se han realizado en el campo de la metodología de desarrollo orientado por pruebas (TDD) y el estándar de calidad de la ISO/IEC 9126; luego se describen los conceptos que son necesarios para la comprensión del proyecto de investigación.

2.1. Estado del arte

La consulta de investigaciones asociadas a productos de software desarrollados bajo el marco de trabajo TDD (Desarrollo orientado por pruebas), que permitan brindar información valiosa en la investigación del presente proyecto, como también las formas de estimar y verificar la calidad del producto utilizando el estándar de la ISO/IEC 9126. Para la identificación del estado del arte del presente trabajo de investigación se tuvieron en cuenta dos temas importantes; el primero fue caracterizar herramientas que involucraban plataformas de información desarrollados bajo la metodología de trabajo TDD (Desarrollo orientado por pruebas), como parte de los procesos de la Ingeniería del Software, y el segundo fue caracterizar los proyectos que involucran el análisis e implementación de la norma ISO/IEC 9126, para evaluar la calidad de un producto de software.

2.1.1. Estudios de proyectos basados en TDD (Desarrollo orientado por pruebas)

A continuación, se muestra una serie de averiguaciones de manera nacional e internacional donde se identifican diferentes estudios y sistemas de software desarrollados utilizando el marco de trabajo de desarrollo orientado por pruebas.

La metodología de desarrollo de software orientado por pruebas (TDD) actualmente es un marco de trabajo que está en proceso de adaptación, existen un sin número de investigaciones, trabajos y proyectos basados en dicho marco de proceso, pero no ha sido adoptada como parte de las empresas. Desde que se lanzó al mundo el modelo de diseño de software, se ha entendido que la base principal de la metodología es la implementación de pruebas unitarias antes de comenzar el desarrollo de la solución. Según [5], el desarrollo orientado por pruebas (TDD) es un modelo que permite a través de sus procesos, crear productos mantenible, de calidad, pero sobre todo mejorar la eficiencia del equipo desarrollador de software. Sin embargo, existen aspectos que se deben tener en cuenta al

momento de implementar el modelo a los procesos de una empresa, debido que al dar los primeros pasos se presentan inconvenientes en cuanto a los costos de capacitación, adaptación del modelo y la productividad del equipo de trabajo, [6] debido a que la metodología de diseño cuenta con un proceso extenso para lograr con el objetivo, es necesario crear una cultura TDD, que se optimicen los procedimientos a medida que transcurre el tiempo, para observar resultados después de realizar cada aspecto lo mejor posible. Es importante mencionar que dicha metodología de trabajo puede ser adoptada por proyectos de diferente tamaño, siempre y cuando se trabajen como proyectos pequeños, es decir módulos separados que puedan ser integrados en las entregas del producto.

La ingeniería del software actualmente es el centro de atención de todas las miradas, como bien se sabe, el mundo gira entorno a las aplicaciones, al software que crean diferentes expertos en la materia; dicha tecnología ha mejorado la calidad de vida de los seres humanos; por eso es importante responder el desafío que supone crear un producto de esta característica, teniendo como base la calidad, los requisitos que solicita el cliente y la difusión de dichas tecnologías. [7] Los modelos agiles es un punto importante de cambio y mejora al proceso, al permitir realizar cambios durante la marcha del desarrollo de un software, debido a que se centra en su totalidad a los requisitos cambiantes por parte del cliente o usuario final del producto, logrando así crear soluciones a la medida. Pero además, existen modelos que buscan mejorar dicho proceso desde el diseño, como el modelo basado en pruebas unitarias antes de iniciar un desarrollo final del producto. Según [8], es bueno contar con herramientas que automatizan el proceso de crear, refinar y verificar las pruebas unitarias que realizan en las fuentes del proyecto, de hecho en la actualidad se han creado distintas herramientas que apoyan y agilizan el proceso para diferentes lenguajes de programación, de manera tal que es algo que está en proceso de adopción, pero que ya lleva un camino amplio en el mercado.

En el transcurso de la presente investigación se ha profundizado en la metodología de desarrollo de software orientado por pruebas (TDD), pero es importante comentar una parte fundamental del desarrollo de dicho modelo y es el marco de trabajo de programación extrema, dicho marco permite la gestión en procesos del desarrollo de software mientras que TDD es un marco de trabajo que se centra en el diseño de un sistema. [9] TDD (Desarrollo orientado por pruebas) fue creado a partir de la metodología XP, es una rama diseñada para optimizar el proceso, mejorar la productividad del equipo desarrollador y obtener productos robustos, mantenible y de calidad; se aplica antes del desarrollo final de las fuentes del proyecto y se centra en las pruebas unitarias de software. Según [10], el marco de trabajo desarrollo orientado por pruebas no ha adquirido la importancia que merece, debido a que, con el transcurso del tiempo se ha podido demostrar que es una buena herramienta de diseño

para ser implementada dentro de los modelos que se utilizan en las empresas, como también es demostrable que dicho marco de trabajo, tiene la capacidad de crear productos de calidad.

Es importante resaltar que el cambio en las empresas que desarrollan software actualmente, se realiza a paso lento, es por esto que la implementación de las metodologías que han ido evolucionando no han sido aprovechas por dichas entidades para avanzar hacia algo que sea realmente eficiente y productivo. [11] En algunas investigaciones realizadas en el campo, muestra que inclusive existen empresas que cuentan en sus procesos los modelos tradicionales, no han podido impulsar un modelo innovador y profesional para ser competentes en el mercado actual. Según [12], es importante contar con un grupo especializado y actualizado en las entidades para investigar las formas posibles de migrar hacia las últimas tecnologías y metodologías; también es importante que el grupo investigador desarrolle nuevos elementos que permitan avanzar no solo a su entidad, sino que sea un aporte amplio, que sea para todas las empresas del mundo, por esto es importante difundir la información y procurar avanzar en la implementación de dichas innovaciones.

La implementación de la metodología de desarrollo guiado por pruebas (TDD), hoy en día cuenta con un conjunto de detractores que argumentan, que dicho marco de trabajo no es posible ser adoptado en proyectos grandes, debido a la complejidad que tiene el proceso algorítmico de dicha metodología; sin embargo, otro conjunto de investigadores de TDD, explican que un proyecto grande, es un conjunto de sub proyectos, que se deben gestionar, partiendo del refrán de divide y vencerás, aplicando el marco de trabajo a cada sub proyecto. Para gestionar adecuadamente el marco de trabajo TDD, es necesario implementar un modelo ágil como SCRUM o Programación Extrema, para trabajar de la forma correcta el backlog del producto y asignar cada tarea a los grupos de trabajo del proyecto. La mayoría de las investigaciones muestran que el desarrollo orientado por pruebas (TDD) tiene beneficios importantes para ser adoptado por las empresas que desarrollan software; de acuerdo a entrevistas que se comentan en la investigación [13] la metodología TDD permite crear código de calidad, debido a que escribir fuentes de forma simple y limpia como fase inicial permite conocer ampliamente la lógica de negocio; TDD permite aplicar modelos de calidad y busca crear un hábito en los desarrolladores de entender el sistema, antes de comenzar e escribir líneas de código sin saber el rumbo del proyecto; TDD aumenta la productividad de los equipos de desarrollo debido a que a medida que se realiza el proceso algorítmico, con el tiempo dicho equipo aumenta la producción y como resultado la comunicación de los involucrados de cada proyecto.

Las metodologías agiles que se han creado a través del tiempo, siempre han sido sometidas a todos tipos de cuestionamientos, análisis, estudios y simulaciones para determinar el nivel

de confiabilidad en cuanto a la calidad de los productos y las ventajas que tiene el modelo en relación al costo beneficio. De acuerdo a la documentación, la metodología de desarrollo de software orientado por pruebas (TDD) tiene varias investigaciones y simulaciones que permiten mostrar los pros y contras de la adopción de dicho modelo en las empresas que desarrollan software. Según la investigación [14] al realizar la simulación de varios tipos de proyecto de software que incorporan la metodología TDD y equipos de trabajo que han adoptado el marco de trabajo TDD en sus procesos, describen al modelo como una herramienta eficiente en la utilización en las empresa, de acuerdo a cualquier tipo de características, requisitos o tecnologías que solicite el proyecto. El desarrollo de cualquier proyecto bajo el marco de trabajo TDD, debe contar con el conocimiento de la utilización de los procesos que define la metodología, es importante que el equipo de trabajo sea experto en el tema para aumentar la productividad en la creación de un producto, debido a que tal vez es la falencia más importante que tiene el modelo cuando es adoptado por las empresas que desarrollan software.

La metodología de desarrollo de software orientado por pruebas, es un marco de trabajo que ha ido adquiriendo fuerza a través del tiempo, diferentes investigadores le han querido dar la importancia de la implementación de dicho modelo para ser adoptado, en las empresas que desarrollan software; dentro de este conjunto de innovadores, hay otro conjunto de curiosos que están buscando la forma de transmitir el conocimiento al crear técnicas, didácticas y tecnologías para caracterizar y explicar a profundidad la teoría que hay detrás del marco de trabajo, los beneficios que trae la implementación del modelo en las empresas, la optimización de los procesos y el conocimiento que debe tener el equipo desarrollador para aplicar de la mejor manera dicha metodología. Según la investigación [15] es necesario crear herramientas que permitan enseñar la metodología de trabajo orientado por pruebas (TDD) a cualquier persona, empresa o entidad que desee profundizar en dicho marco de trabajo, también es importante que docentes investiguen e infundan información para que los estudiantes y personas logren sembrar esa semillita del interés por querer aprender las últimas tecnologías para beneficio de sí mismo y la organización.

2.1.2. Modelos de calidad según el estándar ISO/IEC 9126

A continuación, se muestra un conjunto de indagaciones por parte de investigadores a nivel nacional e internacional donde se identifican diferentes estudios en la implementación del estándar de calidad de la ISO/IEC 9126.

El modelo de calidad del estándar de la ISO/IEC 9126 brinda un amplio abanico de características que se pueden evaluar de un producto de software totalmente terminado, el

modelo muestra de una forma objetiva el cumplimiento de requisitos del sistema de una aplicación [16]. Al igual que las metodologías agiles, existen diferentes modelos de calidad que permiten evaluar los sistemas de información, pero aún están en proceso de desarrollo, debido a que dichos modelos creados se centran en algunos elementos importantes, como por ejemplo en el rendimiento, usabilidad, portabilidad, funcionalidad entre otros, pero no se han integrado en una metodología que abarque la verificación de las características de un software desde el diseño del producto hasta la puesta en producción, mantenimiento y soporte de dicho proyecto. Existen diferentes metodologías de evaluación en el proceso completo de desarrollo de software y como tal el producto final, pero para poder realizar dicha evaluación se deben integrar varios modelos de evaluación de calidad para lograr el objetivo, por lo tanto es un proceso algo ineficiente para las entidades. Según [17], es posible utilizar la norma de la ISO/IEC 9126 para evaluar el proceso completo del ciclo de vida del desarrollo de un software, sino que se deben adaptar las características del modelo de calidad a los procesos que se tienen en cada paso del proceso, es decir evaluar el modelo conceptual de la base de datos, evaluar los prototipos del software, evaluar las historias de usuario ya así sucesivamente hasta completar ampliamente todos los aspectos que se deben tener en cuenta del desarrollo de software.

Actualmente, debido al alto porcentaje de aplicaciones en el mercado, se han diseñado modelos de evaluación de calidad, para ser aplicados en algunos de estos productos, cada uno de estos modelos se han creado con el fin de brindar confiabilidad a los usuarios finales del producto, es importante mostrar a los usuarios productos de calidad para así aumentar el target poblacional y aumentar las utilidades en las empresas. El modelo de calidad de software de la ISO/IEC 9126 es una herramienta determinante en la evaluación de las aplicaciones, que permiten dar confianza de dichos sistemas y mejorar las expectativas de lo que desea ver el cliente. Según el estudio [18], es importante que al momento de realizar una evaluación de algún producto de software, se tengan en cuenta la totalidad de características del modelo de calidad y en lo posible mencionar un conjunto de atributos amplio de lo que es la portabilidad, mantenibilidad, eficiencia, usabilidad, fiabilidad y funcionalidad del producto. Con el auge de la tecnología es relevante que dichos sistemas sean verificados antes de ser lanzados al mercado, e inclusive procurar en lo posible que las herramientas que se despliegan para ayuda del desarrollador, permitan facilitar la evaluación de los productos de software; por ejemplo algo que hoy en día es bastante común en los Frameworks de desarrollo, encontrar las librerías integradas para facilitar la realización de pruebas unitarias.

En algunos países del mundo es obligatorio que se implementen métricas de calidad a las plataformas que tienen alguna relación con el gobierno, inclusive para sitios en internet que son desarrollados por empresas del sector privado; las plataformas como las de hospitales, la

policía, ejercito, gobierno, grupos de investigación, interventorías, organismos de control, fiscalía, ministerios y Universidades tienden a ser regulados por el estado, aplicando normas, que deben ser cumplidas para lograr, llegar al usuario de la mejor manera posible y a un target poblacional mucho mayor; de lo contrario dichas entidades serían multadas por incumplimiento a la norma ya establecida. Sin embargo, hoy en día y por efectos de la misma corrupción se han podido encontrar diferentes plataformas que no cumplen con el mínimo de usabilidad o funcionalidad y cuenta con sin número de fallas posibles, software que ha sido desarrollado por empresas nada competentes, que responden con los contratos creando un producto con muchas falencias, pero que al final son pasadas como si cumplieran con las características del estándar de la ISO/IEC 9126. Según [19], es importante que los gobiernos regulen la calidad de los productos de software para que en realidad sean aprovechados por la población, que si existe una plataforma por ejemplo para solicitar citas médicas en el hospital, que dicho sistema sea lo más usable posible, que personas discapacitadas lo puedan utilizar, que los colores utilizados en la interfaz de usuario sean lo más relajantes en lo posible, que cuente con paneles de ayuda, es decir facilitarle el uso del sistema a los clientes y todo esto se logra implementando el estándar de la ISO/IEC 9126, de forma objetiva en sus diferentes características.

Es importante aclarar que debido al alza en la demanda de los productos de software, es relevante que se destaquen por la calidad, dichos sistemas, por eso es bueno determinar que al igual que el modelo de la ISO/IEC 9126, que permiten valorar las plataformas y aplicaciones desde diferentes puntos de vista, también se hagan uso de modelos como la ISO 9000, ISO 9001, ISO 9004, el estándar de la ISO 27001 y entre otros que brindan confianza y calidad a los usuarios de dichos sistemas; que no solo es bueno para los usuarios finales, sino que también es una meta lograda, un éxito por parte de las empresas. La implementación de dichos estándares también representa organización de los procesos, organización del equipo de trabajo, representa liderazgo, compromiso y responsabilidad por adoptar y mejorar la competitividad en el mercado. Es importante tener en cuenta que al momento de adoptar un elemento que ayuda en los procesos de la empresa se realicen estudios, para que se implementen los modelos más adecuados y no incurrir en gastos innecesarios, cada proceso tiene su tiempo de adaptación y mejora en la organización, por eso es bueno seguir los lineamientos. Según [20], es necesario que se migre a todas este tipo de herramientas, metodologías y estándares para competir en el mundo global de hoy, debido a que en otros países donde la cultura es disciplinada, diferente a la nuestra, es algo que se tiene implementado y es por esto que son países desarrollados que demandan mucho software de calidad.

La norma técnica de la ISO/IEC 9126 es un modelo de calidad que facilita la evaluación de las diferentes vistas que tiene un producto de software, permite trazar los lineamientos con los que debe contar el sistema de información para ser objetivamente competitivo en el mercado, pero que sobre todo, que el diseño metodológico basado en el estándar de calidad, permita desarrollar las actividades y procesos de manera que generen valor significativo en las empresas y llegar al punto máximo de optimización de los procesos y la calidad del producto, en esta situación para las aplicaciones y plataformas. De acuerdo a la investigación [21] el modelo es una pieza fundamental para evaluar los sistemas de información, dispone de un conjunto de características, de las que se pueden tomar para establecer parámetros que permiten analizar aspectos como efectividad, productividad, seguridad y satisfacción, pero dichos aspectos pueden ser modificados o agregados por la metodología de evaluación de diseñe cada organización y que permita analizar objetivamente los productos de software. Es importante informar que la evaluación y competitividad de las empresas, no solo se centran en la ISO/IEC 9126, sino que también se deben complementar dichos procesos, con los demás modelos, estándares y metodologías, para lograr el éxito esperado; es un conjunto de elementos organizados y estructurados que son aprovechados por la organización.

El desarrollo de software actualmente es un abanico difícil de descifrar; el proceso, las herramientas, los clientes son cada vez más exigentes; la adopción de varios estándares, certificaciones, metodologías también son algo costosas para las empresas, es por eso que dichas entidades siempre están en ese proceso de asegurar e invertir para dar credibilidad y calidad a sus productos, pero que inicialmente lleva un esfuerzo valioso y complicado para lograr con el fin de mejorar los procedimientos de la organización. Según [22] las empresas que invierten en la incorporación de tecnologías, certificaciones y metodologías aseguran la calidad del software, dando confiabilidad a sus clientes y las unidades de negocio de la organización. Además la investigación muestra como conclusión, que la mayoría de proyectos de software que tienen éxito es menor en comparación con los productos que fueron cuestionados, sufrieron cambios o no cumplieron con los requisitos del cliente, otro valor importante de los proyectos fracasaron, por tal información es importante adoptar modelos de calidad y metodologías presentes en el mercado, para definir las características viables que definen de la forma correcta de los procesos de la organización y como resultado la satisfacción y efectividad de los productos de software desarrollados.

2.2. Marco conceptual

El marco conceptual del presente proyecto, permite realizar un bosquejo general de las teorías que abordan el tema de investigación, que buscan dar claridad de los conceptos básicos que se deben entender para reconocer de manera amplia el fin de la realización del proyecto.

2.2.1. Arquitectura de software.

La arquitectura de software como parte de la ingeniería de software, facilita el proceso de diseño de un sistema, debido a que desglosa lo diferentes módulos que hacen parte de la totalidad del sistema en diferentes vistas, para dar claridad de los componentes que conforman del producto, las posibles relaciones que se dan [23] , las características que definen el funcionamiento y los diferentes ambientes en los que puede operar las aplicaciones al final del proceso. Según la investigación [24] la arquitectura de software es primordial ser aplicada en los procesos de ingeniería de software para favorecer la evolución del desarrollo de software, conocer las partes mínimas del sistema, plasmar el ambiente de desarrollo y mejorar la calidad de los productos de software.

A continuación, se muestra los aspectos primordiales de la arquitectura de software en el desarrollo de proyectos:

- Permite perfeccionar el entendimiento de los desarrollos de software que tienen procesos engorrosos y extensos.
- Brinda en ambiente de trabajo propicio de los entes involucrados en el desarrollo de los sistemas de software.
- Crea un modelo de diferentes posibilidades en la construcción de métricas y diseños de software para reutilización de elementos de código.
- Muestra con claridad las vistas generales en el desarrollo de software a medida que avanza el proceso.

Determinar la arquitectura de software para un proyecto de software es un proceso complejo al inicio del proyecto, pero es importante que se desarrolle dicho modelo, porque facilita el entendimiento de las tecnologías a usar, los requisitos del sistema funcionales y no funcionales y el modelo de producción de las aplicaciones.

2.2.2. Arquitectura modelo vista controlador

El modelo vista controlador es una arquitectura que simplifica el desarrollo de aplicaciones, debido a que permite desarrollar proyectos escalables, separado por capas y mantenible en

el tiempo [23]. La arquitectura separa el desarrollo de software en tres capas: la del modelo, en esta capa se definen las entidades para el almacenamiento de la información y la lógica de negocio; la vista, en esta capa se visualiza la información registrada y procesada en la capa del modelo, a los usuarios del sistema [25]; y el controlador, en esta capa se realiza todo el procesamiento de la información y la intercomunicación que tiene cada usuario con el sistema. El modelo vista controlador facilita la realización de cambios en las fuentes, debido a que lo que se realice en alguna de ellas no afecta en ningún aspecto a las demás, [26] que es una de las ventajas de utilizar dicha arquitectura y que además posibilita el desarrollo en componentes de software.[27]

La figura 1 muestra la arquitectura MVC en sus elementos y la interacción.

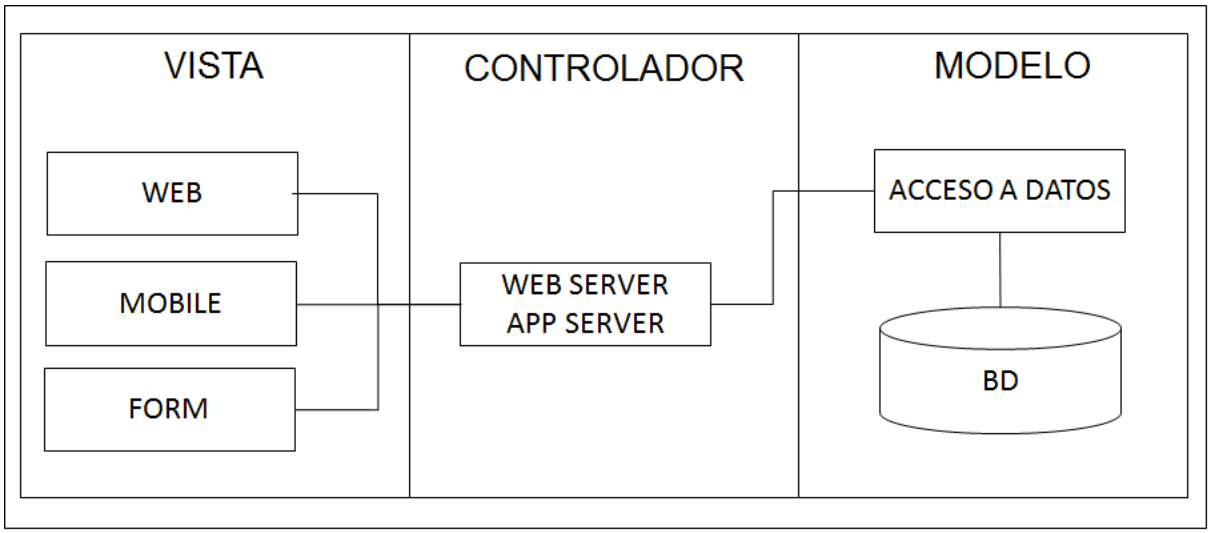


Figura 1: Modelo vista controlador (adaptado [28] *)*

De las ventajas que brinda la arquitectura del modelo vista controlador se pueden mencionar las siguientes [29]

- La separación de forma clara de los componentes del software.
- La conexión entre el modelo y las vistas es dinámica; se produce en tiempo de ejecución, no de compilación.

2.2.3. Metodologías ágiles de desarrollo de software

Los marcos de trabajo para el desarrollo de software surgen en los años 60’s, para proporcionar un estructura de procesos organizada desde las fases tempranas del desarrollo de una aplicación hasta el mantenimiento y soporte del producto. [30]

Las metodologías pioneras en la ingeniería de software son conocidas actualmente como metodologías tradicionales, en el que los procesos se manejaban un poco directos, difíciles

de entender y contaban con una amplia documentación en el diseño de la aplicación, el proceso fue lineal y cada requisito debía ser aprobado por el cliente en dicho proceso, sin contar con modelo anticipado [31]. Al pasar el tiempo y detectar un problemática seria en la calidad del software, debido a que se estaban creando productos bastantes alejados de la realidad de lo que realmente deseaba el cliente, surge en el año 2001 el manifiesto ágil y junto con él, emergen lo que se conoce actualmente como las metodologías agiles, para suplir la necesidad de un nuevo modelo en el diseño y construcción de las aplicaciones. [32] Las metodologías agiles focalizan su importancia en permitir los cambios que solicita el cliente en la evolución del desarrollo del sistema, cada vez que se realiza una entrega del producto, acepta la documentación pero no es relevante para el proceso y es mínima, el conjunto de procesos es visual y el compromiso es grupal del equipo desarrollador [33]. Las metodologías ágiles también se centran más en el bienestar de las personas, por esto la mayoría de los marcos de trabajo enfocados en el manifiesto ágil se preocupan por el buen ambiente, para que el equipo de trabajo logre realizar las actividades de forma eficiente y además elimina la línea de mando de los modelos tradicionales y ubica una figura de liderazgo. [34]

2.2.3.1. Metodología XP

El marco de trabajo XP (Programación extrema) es una perspectiva de la ingeniería de software y de los modelos agiles en la creación de aplicaciones, que se centra básicamente en mejorar la comunicación del equipo de trabajo, en facilitar la conexión de los involucrados del proyecto y propiciar la colaboración. La metodología XP le da su valor en los modelos ágiles al procurar vincular al cliente como parte del equipo de trabajo, para logra entender ampliamente los requisitos del sistema, también respalda el proceso hasta el mantenimiento y soporte del proyecto para mejorar la calidad del producto durante la línea del tiempo de producción. [30]. XP documenta los requisitos del cliente a través de historias de usuario, que son pequeñas unidades que describen un requisito del sistema de lo que desea el cliente en sus propias palabras, pero que además muestran las condiciones que debe tener dicha unidad. La metodología XP brinda facilidad a la hora de asignar los roles, debido a que cada miembro del equipo de trabajo puede desempeñar uno o varios roles, de acuerdo a las condiciones que exige el proyecto de software. [35]

El ciclo de vida del desarrollo de software en la metodología XP, se realiza en seis ciclos que se describen a continuación, la primera es la exploración, en este ciclo se especifican los requisitos del cliente; la planificación de la entrega, se definen las historias de usuario a desarrollar y las posibles entregas del producto; las iteraciones, se realiza el proceso de evolución del producto; la producción, en este ciclo que crea el sistema de información; mantenimiento, se realizan las modificaciones en las que haya inconvenientes y se brinda el

soporte para verificar el correcto funcionamiento del producto y la última etapa es la finalización del proceso del proyecto. [36]

En la metodología XP existe un conjunto de consideraciones a tener en cuenta para que al momento de adoptar el modelo en las empresas, se logre disminuir los posibles costos en los que incurre la empresa en la implementación de dicha metodología:

Es importante propiciar la comunicación de los involucrados del proyecto y el cliente de la plataforma a desarrollar.

Realizar entregas del producto al cliente y promotor del producto en lapsos de tiempo temporales y pequeños.

Es importante enfocarse en los requisitos del sistema planear diseños simples para lograr con el objetivo del proyecto.

La metodología XP propicia el desarrollo de las fuentes del proyecto en dos personas, es decir los dos en un mismo ordenador de trabajo.

2.2.3.2. Metodología TDD (Desarrollo orientado por pruebas)

La metodología TDD (Desarrollo orientado por pruebas) es un marco de diseño e implementación en el desarrollo de software, fue creado a partir de la metodología ágil XP (Programación extrema). [1] TDD centra su importancia en el diseño de software en escribir pruebas unitarias antes de comenzar a desarrollar el sistema y así comprender ampliamente el sistema que se desea crear a final del proceso. El algoritmo que se utiliza en TDD parte desde el planteamiento del requisito del sistema, luego la implementación del código fuente de la prueba que falle como primera medida, luego que pase satisfactoriamente las pruebas del paso anterior, y finalmente refactorizar el código escrito para eliminar la duplicidad y hacer mejoras. Una vez el desarrollador tiene claro la funcionalidad del software, lo expresa en código, al tener el ejemplo escrito, se codifica lo mínimo necesario para que se cumpla y la prueba pase de forma correcta. Finalmente, se modifica el diseño sin alterar su comportamiento, en búsqueda de líneas duplicadas; se revisa que el código cumpla con ciertos compendios de diseño que se deben mantener en el proceso de la ingeniería de software, para así determinar la aptitud de las fuentes del proyecto. [37]

Ciclo de vida: el ciclo de vida que propone la metodología consiste en los siguientes pasos: el cliente escribe su historia de usuario en un documento formal, luego se escriben los criterios de aceptación de cada historia de usuario junto al cliente, se desglosan cada una de estas características y simplifican en lo permitido, se selecciona el criterio de conformidad más

simple y se convierte en una prueba unitaria en las fuentes del proyecto, se comprueba que la prueba falla en la primera verificación, luego se corrige la prueba unitaria para que pase la verificación, se ejecutan todas las pruebas automatizadas, se refactoriza y se limpia el código, se vuelven a pasar todas las pruebas automatizadas para comprobar que todo sigue funcionando, se vuelve con las premisas de evaluación que falten y se repite el ciclo una y otra vez hasta terminar el software. [38]

De acuerdo al punto de vista de Kent, su creador, TDD tiene los siguientes beneficios al utilizar la metodología:

- La calidad del producto de software aumenta, porque se elimina duplicidades y se comprende ampliamente desde los pasos iniciales.
- Se consigue código altamente reutilizable, debido a que se crean funcionalidades altamente reutilizables en diferentes clases.
- El trabajo entre el equipo desarrollador se hace más cómodo y facilita la relación de las personas
- Propicia que el equipo de trabajo tenga la facilidad de confianza en cada uno de sus compañeros aunque tengan menos experiencia
- Multiplica la comunicación entre los miembros del equipo
- Las personas encargadas de la garantía de calidad adquieren un rol más inteligente e interesante
- Escribir el ejemplo (test) antes que el código nos obliga a escribir el mínimo de funcionalidad necesaria, evitando sobre diseñar. [39]
- Cuando se revisa un proyecto desarrollado mediante TDD, se deduce que los test son excelentes como documentación práctica que se puede consultar, cuando se requiera entender la pieza de código [40]
- Incrementa la productividad del equipo desarrollador.
- Descubre y afronta más casos de uso en tiempo de diseño del software.

2.2.4. Modelo de calidad de los sistemas

Los modelos de calidad para el software son características definidas en un estándar que posibilitan medir el software de acuerdo a un nivel de eficiencia deseado por el desarrollador y los usuarios finales del sistema. Las métricas de calidad del software analizan desde diferentes puntos de vista la aplicación, de si esta cumple o no con los requisitos funcionales especificados en las historias de usuario y los no funcionales tomados desde la experiencia de la organización. Las características están descritas en las particularidades de eficiencia en la resolución de problemas, la flexibilidad para adaptarse al ambiente de producción, la

confiabilidad, determinar si es una aplicación segura o no; mantenibilidad, definir el nivel de tolerancia a fallos; portabilidad, que tenga la capacidad de funcionar en las diferentes plataformas y en diferentes navegadores en caso de ser un sitio web; y la usabilidad que básicamente se centra en la facilidad de aprendizaje para operar el sistema. [41]

Es importante señalar que existen en los diferentes medios, metodologías, estándares y filosofías que apoyan la calidad del producto desde el proceso; es decir favorecen la evaluación de las actividades que se realizan desde el diseño, pruebas, desarrollo, mantenimiento y soporte de las aplicaciones, para que en el cierre de dichos proyectos, se obtengan productos realmente de calidad. [42]

Para el presente proyecto se utilizó el modelo de calidad de la ISO/IEC 9126, el cual es un estándar, que evalúa las aplicaciones web desde diferentes puntos de vista, facilitando un conjunto de métricas que se pueden estimar cualitativamente o cuantitativamente para medir el nivel óptimo de los productos de software. El estándar de calidad reúne características que valoran los procesos desde la perspectiva interna y externa de las plataformas web.[43]

2.2.5. Pruebas unitarias

Las pruebas unitarias son pequeñas unidades (métodos) de código de un módulo de software, debidamente estructurados, que permite verificar la funcionalidad en la mínima expresión de las funciones que debe tener el sistema. Las pruebas unitarias son parte del diseño de un software, que especifican los requisitos del código y ayudan a verificar sus resultados. El proceso que conlleva desarrollar una prueba unitaria, comienza desde la definición de requisitos del sistema (historias de usuario), el siguiente paso se centra en la codificación de la prueba unitaria, donde se realizan las respectivas validaciones y verificaciones, para determinar que la funcionalidad creada del requisito del cliente cumpla con lo planteado y finalmente se corrige los inconvenientes presentados, se eliminan los códigos redundantes, las duplicidades y cualquier falla para agrupar las funciones en un conjunto de pruebas unitarias. [44]

Las pruebas unitarias son automatizables, es decir, que los lenguajes de programación, Frameworks y tecnologías, existen herramientas que permiten validar una prueba unitaria desde el mismo código fuente, sin necesidad de realizarlas manualmente. Se pueden realizar un sin número de pruebas unitarias, de tal forma que el diseño de la especificación de las historias de usuario sean lo más completo posible. [45] En muchas de las situaciones, la realización de pruebas unitarias agiliza el proceso de construcción de un sistema de software, pero también permiten ser reutilizadas en diferentes proyectos, para avanzar rápidamente a medida que se aplica el proceso. Las pruebas unitarias es escriben de forma independiente

de las fuentes del proyecto, no interfieren en la solución definitiva del sistema. Algunos de las investigaciones aseguran que la implementación de pruebas unitarias es un proceso demasiado lento, pero realmente crear un código de prueba unitaria, no debe llevar más de cinco (5) minutos y así realizar el proceso de forma ágil.

2.2.5.1. Ventajas de aplicar pruebas unitarias

A continuación se describen algunas de las ventajas importantes de implementar pruebas unitarias al desarrollo de un software:

- Las pruebas unitarias facilitan la reestructuración del código y favorece la integración de los diferentes módulos del sistema; el poder realizar cambios a pequeñas unidades no afecta al total del sistema y por el contrario se corrigen las inconsistencias que se van descubriendo a medida que avanza el desarrollo del sistema.[44]

- El desarrollar pruebas unitarias en un máximo de cinco (5) minutos y realizar la verificación de cada una de ellas, permite detectar a tiempo los errores del código y corregir los inconvenientes presentados a tiempo, sin tener que volver al inicio del ciclo de vida de un software, permitiendo así, agilizar los procesos.

- El realizar pruebas unitarias desde el comienzo del proyecto y seguir el procedimiento de la forma correcta permite detectar inconsistencias a tiempo, refactorizar el código y eliminar redundancia en las fuentes, cuando el proyecto finalice se obtiene un producto de calidad y un código limpio y bien organizado.

- A medida que evoluciona el desarrollo de software utilizando las pruebas unitarias permite proporcionar información para futuros proyectos y evitar que los inconvenientes presentados en anteriores desarrollos afecten los nuevos que se desarrollen y sobre todo, también permite adoptar rápidamente las prácticas de las pruebas unitarias o en su defecto la metodología de desarrollo orientado por pruebas, fomentando la cultura de dicho diseño de software.

- Al comenzar el ciclo de vida de desarrollo de software con las pruebas unitarias, permite dar un contexto amplio del sistema, enfocar con claridad el diseño y determinar ampliamente los requisitos que se deben cumplir, para lograr un producto de calidad.

2.2.5.2 Consideraciones de las pruebas unitarias

- Es importante que las pruebas unitarias se desarrollen como parte de una metodología de desarrollo de software, como el modelo TDD, junto con un marco de trabajo para la gestión

del proceso como XP o SCRUM, para realizar organizadamente el proceso; de lo contrario se presentaran inconvenientes en la implementación del llamado diseño de software.

- Las pruebas unitarias se enfocan en aspectos de la lógica de negocio; es un modelo apartado de la integración, rendimiento, la interfaz de usuario y demás aspectos en conjunto del software. Por lo tanto no permite abarcar un contexto amplio del software.

- Es importante que al implementar pruebas unitarias, se realicen lo más realistas y útiles en lo posible, para lograr desarrollar el proceso de forma objetiva. Es necesario que las entradas de datos a cada una de las unidades, sean los más creíbles para crear pruebas unitarias teniendo en cuenta un contexto amplio del desarrollo del sistema o producto final.

2.2.6. Estándar de calidad del Software ISO/IEC 9126

El estándar de la ISO/IEC 9126 es un patrón universal para la estimación de la calidad del Software, que fue establecido originalmente en el año 1991. La norma puntualiza seis características de aplicación; las seis características son divididas en varias categorías, que representan un modelo detallado para la evaluación de cualquier sistema informático.

2.2.6.1. Características Norma ISO/IEC 9126

El paradigma del estándar de la ISO/IEC 9126 establece diez características para la apreciación de la calidad del Software, seis que son comunes para los elementos internos y externos de las aplicaciones. A continuación se describen las características y sub características que define el estándar de los ISO/IEC 9126, que se usarán para evaluar la plataforma de anuncios publicitarios del presente proyecto.

Funcionalidad: Es la aptitud que tiene el software para facilitar los servicios necesarios a los usuarios del sistema y así cumplir satisfactoriamente con los requisitos funcionales que describen los clientes, en el período de existencia del desarrollo del software. [46]

A continuación, se describen las sub características de cada métrica de calidad:

- **Idoneidad:** Permite evaluar si el software desempeña las actividades por las cuales fue desarrollado, define el cumplimiento de requisitos.
- **Exactitud:** Este atributo permite definir si el software obtenido como un desarrollo final, tiene firmeza de lo que se espera en el funcionamiento de dicho sistema.
- **Interoperabilidad:** La característica radica en evaluar si el sistema obtenido puede interactuar con otros sistemas de diferentes plataformas y en distintos contextos.

- **Seguridad:** Determinar el nivel de acceso a los diferentes módulos del software por parte de terceros. Control de acceso a usuarios que no tiene el privilegio de ingresar al sistema.

Fiabilidad: Analizar la capacidad que tiene un producto de software para mantener las ayudas requeridas por el sistema, durante un tiempo establecido y bajo un marco de condiciones definidas por el administrador. [17]

A continuación se describen las sub características:

- **Madurez:** Permite verificar las diferentes fallas que ha tenido el software durante el proceso de desarrollo y si han sido eliminadas durante el tiempo de pruebas del sistema o del uso de la aplicación.
- **Recuperabilidad:** Verificar las funcionalidades del sistema a nivel operativo y sistema de restauración de datos en cado de presentar algún tipo de fallo.[47]
- **Flexibilidad a fallos:** Evaluar el programa, para determinar el nivel de capacidad en el manejo de inconsistencias y recuperación.

Usabilidad: Esta característica permite analizar el nivel de esfuerzo que debe utilizar los usuarios del sistema para el manejo del producto de software.

A continuación se describen las sub características:

- **Aprendizaje:** Permite analizar la complejidad en cuanto a la facilidad de aprender a utilizar el software por parte del usuario.
- **Comprensión:** Evaluar la facilidad que tiene un software para comprender el funcionamiento de los procesos por parte del usuario.
- **Operatividad:** Permite determinar la capacidad que tiene por parte del usuario, la utilización del software, se verificar si tiene mucho o nada de esfuerzo.
- **Atractividad:** Verificar que la interfaz de usuario del sistema, sea llamativa.

Eficiencia: Medir la capacidad del software en cuanto a sus funcionalidades y los requisitos especificados por el usuario, para determinar el nivel de confianza del uso de dicho sistema.

A continuación se describen las sub características:

- **Comportamiento en el tiempo:** Analizar el tiempo de respuesta que tiene el sistema, verificar su rapidez al realizar alguna solicitud.
- **Comportamiento de recursos:** Determinar que el sistema utilice de forma eficiente los recursos de la máquina.

Mantenibilidad: Esta característica permite analizar el nivel esfuerzo necesario para que un sistema pueda adaptarse a las nuevas especificaciones y requisitos del software que son cambiantes. [46]

A continuación se describen las sub características:

- **Estabilidad:** Verificar que el software se puede mantener en funcionamiento a pesar de realizar modificaciones en alguno de sus módulos o tecnologías de operación.
- **Facilidad de análisis:** Evaluar que las herramientas y tecnologías de desarrollo del software son funcionales, con el objetivo de analizar y detectar fallas.
- **Facilidad de cambio:** Analizar y verificar que el sistema pueda ser modificado en sus módulos.
- **Facilidad de pruebas:** Evaluar que al sistema, se le puedan realizar pruebas de manera fácil.

Portabilidad: Esta propiedad define la capacidad que tiene el software, para ser ejecutado en diferentes plataformas o sistemas operativos. [48]

A continuación se describen las sub características:

- **Capacidad de instalación:** Evaluar el nivel de facilidad de instalación que tiene el software en las diferentes plataformas, donde pueda ser ejecutado.
- **Capacidad de reemplazamiento:** Determinar el esfuerzo requerido para reemplazar el software que se tiene, por otro que cumpla las mismas funciones.
- **Adaptabilidad:** Evaluar que el software desarrollado se pueda trasladar a otras plataformas y que se pueda adaptar fácilmente.
- **Co existencia:** Evaluar que el software pueda ser ejecutado en otros sistemas operativos y convivir con las demás aplicaciones.

3. Objetivos

En este capítulo se describe el objetivo general y los objetivos específicos del presente proyecto, que plasman las metas claras a desarrollar durante todo el proceso de exploración. El objetivo general describe el resultado de la investigación y los objetivos específicos las diferentes metas a lograr, para dar cumplimiento del objetivo general, que se plantean a continuación:

3.1. Objetivo general

Desarrollar una aplicación web de anuncios publicitarios (spot.co), utilizando el modelo de calidad de la ISO/IEC 9126, desarrollada bajo la metodología TDD (desarrollo orientado por pruebas).

3.2. Objetivos específicos

- Desarrollar aplicación web de anuncios publicitarios, utilizando la metodología ágil de desarrollo de software TDD (Desarrollo de software orientado por pruebas).
- Diseñar modelo de calidad de software a aplicar a la plataforma web de anuncios publicitarios, basado en el modelo de calidad de la ISO/IEC 9126.
- Evaluar aplicación web de anuncios publicitarios, bajo el modelo de calidad de software del estándar de la ISO/IEC 9126, teniendo en cuenta las métricas del modelo
- Documentar resultados y conclusiones del resultado de la evaluación de la plataforma web de anuncios publicitarios.

4. Metodología del trabajo

La metodología en la que se fundamenta el desarrollo del proyecto, se agrupa en las principales destrezas para el desarrollo de software determinadas por los marcos de trabajo de SCRUM, XP (programación extrema), TDD (desarrollo orientado por pruebas) y el estándar de calidad de la ISO/IEC 9126.

4.1. Metodología de la investigación

Para la metodología, que se siguió durante el proceso de investigación, se tuvo como base la gestión de la innovación, partiendo desde la revisión bibliográfica para determinar la idea a consultar, el planteamiento del problema, el planteamiento de objetivos, el desarrollo de la investigación y las conclusiones, el proceso de describe claramente a continuación:

- **Fase 1:** Revisión, recopilación y clasificación de la información.
- **Fase 2:** Planteamiento del problema.
- **Fase 3:** Ilustración de objetivos
- **Fase 4:** Desarrollo de la investigación
- **Fase 5:** Conclusiones

La metodología de desarrollo se describe a continuación teniendo en cuenta las anteriores fases y así dar cumplimiento al proceso de investigación del presente proyecto:

En la primera fase se realizó una revisión de las diferentes teorías, que existen en los diferentes medios, en lo que respecta al marco de trabajo de desarrollo de software orientado por pruebas (TDD), el estándar de calidad del ISO/IEC 9126 y la diferente información en lo que corresponde a la construcción de software, la programación extrema (XP), las pruebas unitarias y la implementación de los modelos de calidad y así recopilar y clasificar dicha información y obtener un marco teórico y conceptual, bien fundamentado y sentar las bases de la investigación. La revisión de la información se tuvo en cuenta diferentes investigaciones realizadas a nivel nacional (Colombia) e internacional para conocer de la mano el avance de nuestro país en materia de las metodologías agiles. Al contar con el marco teórico se procedió a definir el problema a solucionar en la presente investigación. La problemática se centra en la baja utilización de los modelos agiles en las empresas que desarrollan software, por eso es importante profundizar en un modelo, para determinar la viabilidad de ser implementados en la organización. En la siguiente fase, se plasman los objetivos generales y específicos para

desarrollar el proyecto de investigación. En la siguiente fase se describe el proceso de desarrollo de la investigación que se centra primeramente en el desarrollo de un sitio web de anuncios publicitarios utilizando la metodología de desarrollo de software orientado por pruebas (TDD), el diseño del modelo de calidad basado en el estándar de la ISO/IEC 9126 y la implementación de dicho modelo a la plataforma de anuncios publicitarios (spot.co) para determinar el nivel de confianza que tiene el marco de trabajo TDD. Por último se procede a realizar la documentación de las conclusiones de la investigación.

4.2. Fases del ciclo de vida del proyecto

A continuación se describen las fases que se tienen en cuenta para el desarrollo del proyecto de software, teniendo en cuenta las tareas concretas que se desarrollaron desde el planteamiento de la idea del proyecto, la consulta de la información, el desarrollo de la metodología de trabajo del producto de software, la evaluación del modelo de calidad de la ISO/IEC 9126 y las conclusiones del proyecto:

4.2.1. Inicio

Se redacta la propuesta del proyecto. En dicha propuesta se describe el planteamiento del problema, la justificación y los objetivos del proyecto. El documento es sometido a evaluación por parte de la Universidad para determinar la viabilidad del desarrollo de la propuesta de investigación. Al final se afina el planteamiento redactad y se comienza con la planificación del proyecto.

4.2.2. Planificación

Se describe concretamente el planteamiento, las bases teóricas que soportan el proyecto, los objetivos y la metodología para desarrollar el trabajo de grado. En esta fase, se valida la propuesta de la fase de inicio, se procede a realizar la investigación teórica, se diseña el modelo conceptual, lógico y físico de la información y se crean los prototipos del software.

4.2.3. Ejecución

Se identifican los requisitos del sistema, se implementa el marco de trabajo de desarrollo orientado por pruebas (TDD), se utiliza la metodología SCRUM y Programación Extrema (XP) para la gestión del proceso de desarrollo del software y las pruebas. En la identificación de las obligaciones del sistema, se analizan los requisitos, que tiene una plataforma de anuncios publicitarios, el componente tecnológico y los roles del sistema.

Se diseña el modelo entidad relación de la base de datos, a partir del modelo conceptual, se refinan los prototipos de software y se procede a crear el software. Para el desarrollo del sistema, se implementa el modelo algorítmico de la metodología de desarrollo orientado por pruebas (TDD), partiendo desde la creación de las pruebas unitarias que fallan, las pruebas correctas y la refactorización del código de cada una de dichas pruebas. Se implementan las funcionalidades creadas en las pruebas unitarias, a las fuentes del proyecto, se crean vistas y controladores para finalmente obtener el proyecto final. Finalmente, para terminar la solución planteada, se realiza una evaluación del software teniendo como base las características del modelo de calidad de la ISO/IEC 9126. También es importante realizar pruebas de aceptación para verificar el cumplimiento de los requisitos de la plataforma de anuncios publicitarios teniendo en cuenta las historias de usuario.

Durante la fase de desarrollo del proyecto se realizará periódicamente, un control de seguimiento de los adelantos que se realicen en la investigación, de acuerdo a las especificaciones de la metodología planteada.

4.2.4. Cierre

Proporcionar los entregables del proyecto, la sustentación y el cierre formal del proyecto. En la fase final, también es importante el desarrollo de un artículo de investigación como medio de difusión de los resultados del proyecto. En la documentación que se tiene, para la entrega formal del proyecto, se tienen en cuenta las lecciones aprendidas, los resultados, el cumplimiento de objetivos, las conclusiones y las diferentes líneas de investigación que pueden ser abordadas en proyecto futuros.

4.3. Organización del proyecto

De acuerdo a la anterior descripción se presenta la siguiente tabla, para mostrar la distribución de las actividades conforme a las fechas de entrega del proyecto definidas por la Universidad.

Tabla 1: Organización del proyecto

No	Fase del proyecto	Actividad	Ejecución en meses
1	INICIO	**Propuesta**	0,5
		Redactar formulario de solicitud	
		Revisión y aprobación de la propuesta	
2	PLANEACIÓN	**Planteamiento**	1
		Justificación	
		Planteamiento del problema	
		Estructura del proyecto	

3		**Contexto y estado del arte**	
		Introducción	
		Base teórica	
		Conclusiones	
4		**Objetivos concretos y metodología de trabajo**	
		Objetivos	
		Metodología del trabajo	
5	**EJECUCIÓN**	**Desarrollo específico de la contribución**	**2**
		Identificación de requisitos	
		Diseño de base de datos	
6		**Ejecución**	
		Desarrollo del software	
		Evaluación	
7		**Verificación, pruebas y validación**	
		Preparar pruebas y realizar verificación	
		Preparar y evaluar el producto y sus componentes	
8		**Conclusiones y trabajo futuro**	
		Conclusiones	
		Líneas de trabajo futuro	
9	**SEGUIMIENTO**	**Informe del proyecto**	**0,5**
		Entrega plan de trabajo	
		Entrega de la estructura en capítulos	
		Entrega del borrador del TFM	
		Borrador final para revisión	
		Depósito TFM	
10	**CIERRE**	**Cierre del proyecto**	**0,5**
		Sustentación	

Fuente: autor

La tabla 1, es utilizada como medio de seguimiento en la gestión del proyecto, teniendo en cuenta el tiempo para el adelanto de la investigación, hasta la entrega final del proyecto.

5. Desarrollo de la investigación

En este capítulo se describen el desarrollo de la plataforma de anuncios publicitarios (spot.co), los modelos de ingeniería del software, prototipos, implementación de metodología de desarrollo orientado por pruebas (TDD), el desarrollo del sitio web y la evaluación de calidad de la plataforma utilizando el estándar ISO/IEC/9126. Esto suele verse reflejado en la siguiente estructura de apartados:

5.1. Diseño y construcción del software

En este capítulo se describe la lógica de negocio, para el desarrollo del sistema de anuncios publicitarios. Para el desarrollo de la plataforma se debe tener en cuenta las historias de usuario, los roles del sistema y los modelos de ingeniería del software.

5.1.1. Definición de historias de usuario

En el ciclo de vida del desarrollo de un proyecto de software el primer paso a concretar con el cliente son las funcionalidades que debe tener el sistema. La metodología XP y SCRUM se centran en la escritura de pequeñas unidades conocidas como historias de usuario. Las funciones del sistema se determinaron a partir de ejemplos y revisión de sitios en internet de lo que debe tener una plataforma de anuncios publicitarios. Todas las historias de usuarios fueron plasmadas teniendo en cuenta los requisitos que caracterizan a un sitio web de anuncios publicitarios.

Tabla 2: **Historia de usuario:** HU01 – agregar anuncio

Historia de usuario – Sistema de información spot.co	
Número: HU01	**Nombre**: Agregar anuncio
Descripción: el sistema permitirá registrar nuevos anuncios en la plataforma, para que se puedan visualizar en el sitio web. Los datos que se deben proporcionar para un nuevo anuncio son: ▪ Fecha de publicación ▪ Fecha de terminación ▪ Tipo de Anuncio ▪ Descripción ▪ Sucursal ▪ Categoría	

Observaciones: para registrar un anuncio en el sistema, el usuario debe estar autenticado como *Administrador*.

Fuente: autor

Tabla 3: **Historia de usuario:** HU02 – ver anuncios

Historia de usuario – Sistema de información spot.co	
Número: HU02	**Nombre**: Ver anuncios
Descripción: el sistema permitirá visualizar una lista de los anuncios activos de cada sucursal. Se mostrará información de fecha de publicación, fecha terminación, tipo, descripción, sucursal y la categoría a la que aplica.	
Observaciones	

Fuente: autor

Tabla 4: **Historia de usuario:** HU03 – actualizar anuncio

Historia de usuario – Sistema de información spot.co	
Número: HU03	**Nombre**: Actualizar anuncio
Descripción: el sistema permitirá actualizar la información de cada anuncio. Los datos que se pueden editar son fecha de publicación, fecha de terminación, tipo, descripción y sucursal.	
Observaciones: para actualizar un anuncio en la plataforma, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 5: **Historia de usuario:** HU04 – deshabilitar anuncio

Historia de usuario – Sistema de información spot.co	
Número: HU04	**Nombre**: Deshabilitar anuncio
Descripción: el sistema permitirá deshabilitar un anuncio y la información que este asocie.	
Observaciones: para deshabilitar un anuncio del sistema, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 6: **Historia de usuario:** HU05 – agregar contacto

Historia de usuario – Sistema de información	
Número: HU05	**Nombre**: Agregar contacto spot.co
Descripción: el sistema permitirá agregar información de contacto de las sucursales de las empresas. Los datos que se deben registrar están clasificados así: ▪ Descripción	

▪ Tipo ▪ Sucursal
Observaciones: Antes de ingresar los datos de un nuevo contacto se debe verificar que ya se tengan registros de sucursales de cada empresa, de no ser así ya es necesario registrarla. Para registrar un contacto en el sistema, el usuario debe estar autenticado como *Administrador*.

Fuente: autor

Tabla 7: **Historia de usuario:** HU06 – ver contacto

Historia de usuario – Sistema de información spot.co	
Número: HU06	**Nombre**: Ver contacto
Descripción: el sistema permitirá visualizar la información de los contactos que se registren de cada una de las sucursales de las empresas.	
Observaciones:	

Fuente: autor

Tabla 8: **Historia de usuario:** HU07 – actualizar contacto

Historia de usuario – Sistema de información spot.co	
Número: HU07	**Nombre**: Actualizar contacto
Descripción: el sistema permitirá actualizar la información general de los contactos de las sucursales de cada empresa.	
Observaciones: para actualizar un contacto en el sistema, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 9: **Historia de usuario:** HU08 – agregar sucursal

Historia de usuario – Sistema de información spot.co	
Número: HU08	Nombre: Agregar sucursal
Descripción: una sucursal puede participar en uno o más contactos, por lo tanto es necesario registrar información de esta para cada uno. Los datos que se deben registrar son: ▪ Dirección de la entidad ▪ Empresa ▪ Lugar	
Observaciones: para registrar una sucursal de una empresa en el sistema, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 10: **Historia de usuario:** HU09 – actualizar sucursal

Historia de usuario – Sistema de información spot.co	
Número: HU09	**Nombre**: Actualizar sucursal
Descripción: el sistema permitirá actualizar los datos de las sucursales de cada empresa registradas en el sistema.	
Observaciones: para actualizar los datos de una sucursal, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 11: **Historia de usuario:** HU10 – ver sucursal

Historia de usuario – Sistema de información spot.co	
Número: HU10	**Nombre**: Ver sucursal
Descripción: el sistema permitirá visualizar los datos básicos de las sucursales de cada empresa en el sitio web.	
Observaciones	

Fuente: autor

Tabla 12: **Historia de usuario:** HU11 - deshabilitar sucursal

Historia de usuario – Sistema de información spot.co	
Número: HU11	**Nombre**: Deshabilitar sucursal
Descripción: El sistema permitirá deshabilitar una sucursal y los datos que esta asocie.	
Observaciones: para deshabilitar una sucursal el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 13: **Historia de usuario:** HU12 – agregar empresa

Historia de usuario – Sistema de información spot.co	
Número: HU12	**Nombre**: Agregar empresa
Descripción: el sistema permitirá el registro de las empresas que publicarán los anuncios en el sitio web, de los cuales se requiere almacenar la siguiente información: ▪ Nombre de la empresa ▪ Imagen de la empresa ▪ Slogan ▪ Fecha de constitución de la empresa ▪ Categoría	
Observaciones: Para registrar una empresa en el sistema, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 14: **Historia de usuario:** HU13 – ver empresas

Historia de usuario – Sistema de información spot.co	
Número: HU13	**Nombre**: Ver empresas
Descripción: el sistema permitirá visualizar información de las empresas. La plataforma dispondrá un módulo web para mostrar a modo de publicidad la información de la empresa.	
Observaciones:	

Fuente: autor

Tabla 15: **Historia de usuario:** HU14 – actualizar empresas

Historia de usuario – Sistema de información spot.co	
Número: HU14	**Nombre**: Actualizar empresas
Descripción: el sistema permitirá actualizar la información de las empresas registradas en el sistema.	
Observaciones: para actualizar los datos de una empresa el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 16: **Historia de usuario:** HU15 – deshabilitar empresas

Historia de usuario – Sistema de información spot.co	
Número: HU15	**Nombre**: Deshabilitar empresas
Descripción: el sistema permitirá deshabilitar una empresa y la información que ésta asocie.	
Observaciones: para deshabilitar los datos de una empresa el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 17: **Historia de usuario:** HU16 – agregar categorías

Historia de usuario – Sistema de información spot.co	
Número: HU16	**Nombre**: Agregar categorías
Descripción: el sistema permitirá registrar categorías, en las que se puede clasificar las actividades a publicar en los anuncios. Los datos a almacenar son los siguientes: ▪ Nombre de la categoría	
Observaciones:	

Fuente: autor

Tabla 18: **Historia de usuario:** HU17 – ver categorías

Historia de usuario – Sistema de información spot.co	
Número: HU17	**Nombre**: Ver categorías

Descripción: el sistema permitirá ver las categorías registradas en el sistema de información
Observaciones:

Fuente: autor

Tabla 19: **Historia de usuario:** HU18 – deshabilitar categorías

Historia de usuario – Sistema de información spot.co	
Número: HU18	**Nombre**: Deshabilitar categorías
Descripción: el sistema permitirá deshabilitar las categorías registradas en el sistema**.**	
Observaciones: para deshabilitar las categorías el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 20: **Historia de usuario:** HU19 – agregar lugares

Historia de usuario – Sistema de información spot.co	
Número: HU19	**Nombre**: Agregar lugares
Descripción: el sistema permitirá registrar lugares en el portal para localizar la empresa. Los datos a almacenar son los siguientes: ▪ Nombre del lugar ▪ Tipo del lugar ▪ Ubicado en	
Observaciones: para gestionar los lugares del sistema**,** el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 21: **Historia de usuario:** HU20 – actualizar lugares

Historia de usuario – Sistema de información spot.co	
Número: HU20	**Nombre**: Actualizar lugares
Descripción: el sistema permitirá actualizar la información de los lugares registrados.	
Observaciones: para gestionar las actualizaciones a la información de los lugares el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Tabla 22: **Historia de usuario:** HU21 – ver lugares

Historia de usuario – Sistema de información spot.co	
Número: HU21	**Nombre**: Ver lugares
Descripción: el sistema permitirá visualizar en una lista, los lugares registrados en el portal.	

Observaciones: para **visualizar los lugares,** el usuario debe estar autenticado como *Administrador*.

Fuente: autor

Tabla 23: **Historia de usuario:** HU22 – deshabilitar lugares

Historia de usuario – Sistema de información spot.co	
Número**:** HU22	**Nombre**: Deshabilitar lugares
Descripción**:** el sistema permitirá deshabilitar lugares del sistema.	
Observaciones: para deshabilitar lugares el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Al definir las historias de usuario, la siguiente característica a determinar de la investigación son las tecnologías de desarrollo del software y los roles del sistema.

5.1.2. Descripción de los usuarios del sistema

La información administrada por el sistema se centra básicamente en la publicación de anuncios publicitarios de cada ciudad, de tal forma que los usuarios con los que cuenta la plataforma de anuncios publicitarios es un usuario administrador que tiene que iniciar sesión en el sistema y el usuario invitado. A continuación, se muestran los roles:

Tabla 24: Roles de usuario

Rol	Conocimientos	Actividades
Administrador	Usuario con conocimiento básico de ofimática y manejo de cuentas de correo electrónico.	Administrar los módulos y usuarios del sistema de información. Necesita autenticarse para ingresar al sistema.
Invitado	Usuario con conocimiento básico de ofimática y manejo de cuentas de correo electrónico.	Visualizar la lista de anuncios publicitarios. No necesita registrarse en el sistema.

Fuente: autor

5.1.3. Tecnología para el desarrollo de la plataforma

De acuerdo a los diseños de la plataforma de anuncios publicitarios, respecto a la utilización de la arquitectura del modelo vista controlador, los requisitos del sistema y el aprovechamiento

de las herramientas para realizar las pruebas unitarias del proyecto se determinó utilizar las siguientes tecnologías:

Tabla 25: Componente tecnológico

Nombre	Descripción
Python	Lenguaje de programación
HTML	Lenguaje de marcas
CCS3	Estilos en cascada del HTML
Django	Framework para Python
MySQL	Base de datos

Fuente: autor

5.2. Modelo conceptual

El modelo conceptual del diseño de la base de datos del proyecto, define las entidades, atributos y relaciones que determinan el almacenamiento de la información de la plataforma de anuncios publicitarios (spot.co). A continuación, se muestra el modelo:

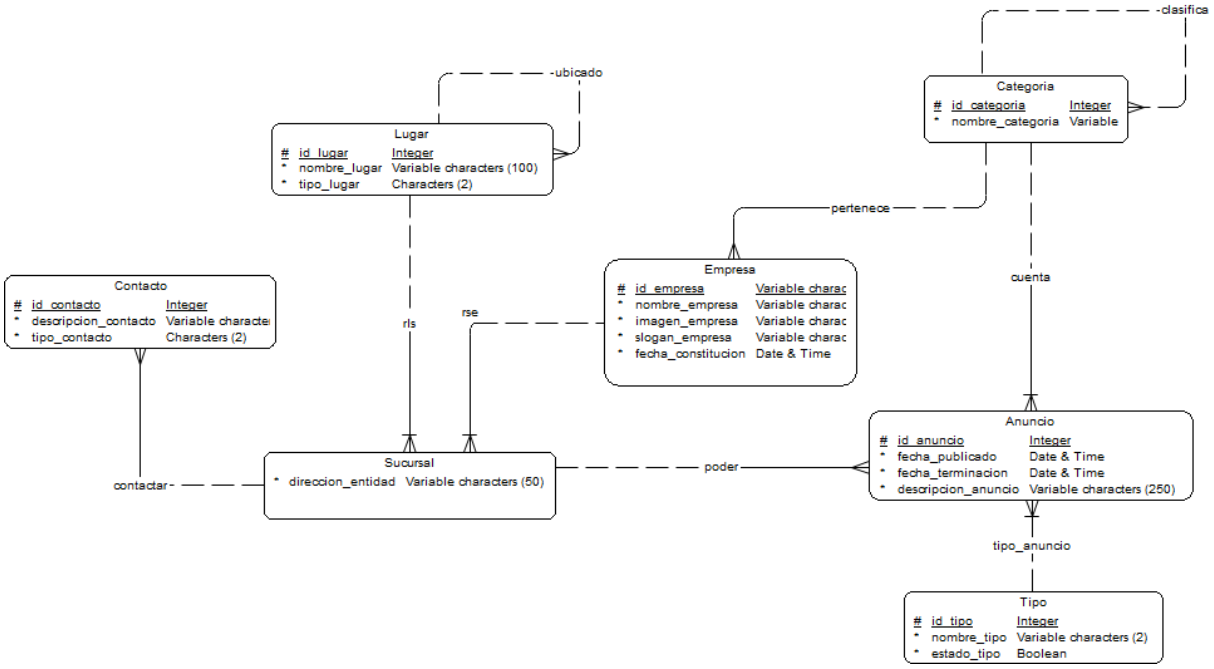


Figura 2. Modelo entidad - relación (autor)

El sistema debe registrar los anuncios publicitarios de las sucursales de las empresas. Cada anuncio asocia un tipo de publicación, una categoría y la sucursal de la empresa, de la cual se desea publicar la información publicitaria. Las categorías representan las actividades

económicas a promocionar dentro del sistema web. Además el sistema debe registrar la ubicación de la sucursal de la empresa y la información de contacto de dicha entidad.

5.3. Diagrama de clases

El diagrama de clases es un modelo importante en la ingeniería de software, debido a que, en él, se plasman las clases, atributos, métodos y relaciones que definen en su totalidad las características que debe tener las fuentes del proyecto. A continuación, se muestra el modelo:

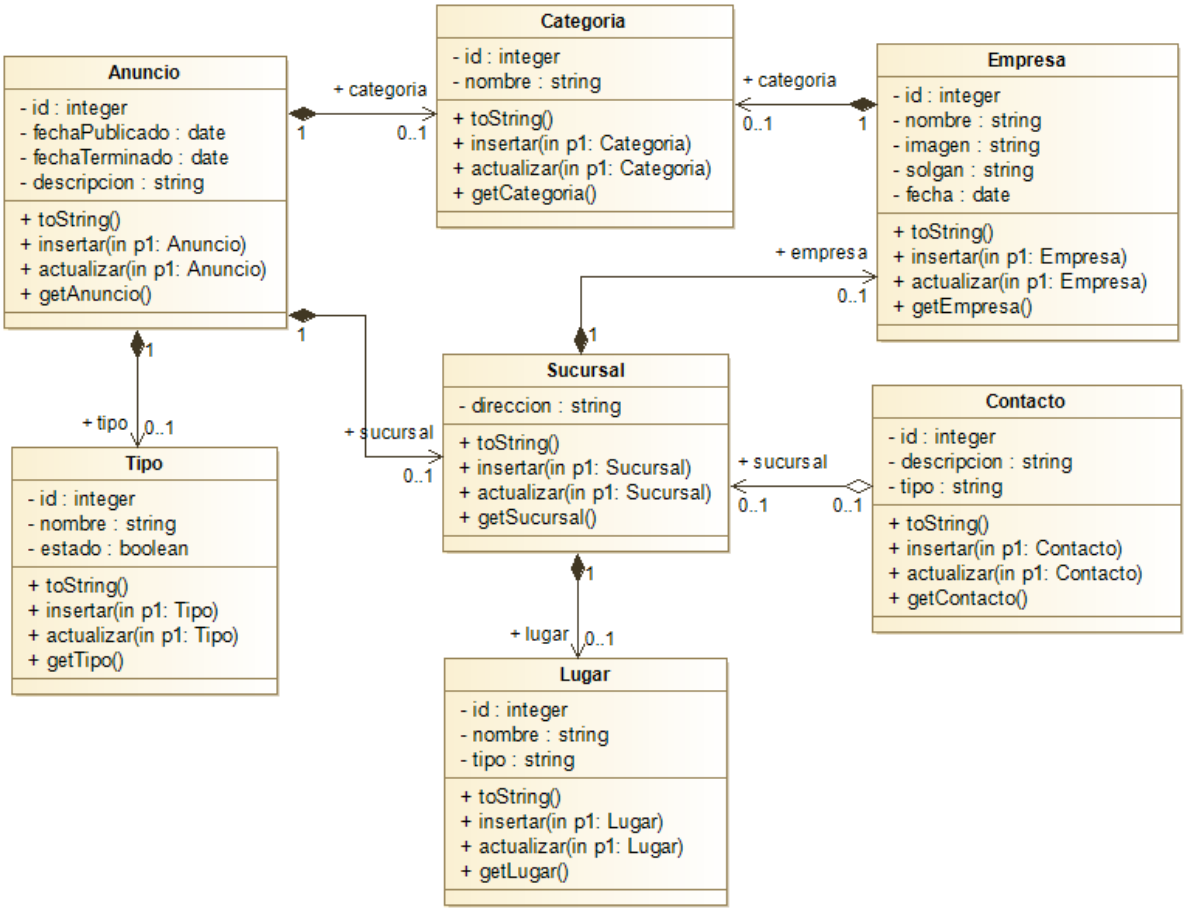


Figura 3: Modelo de clases (autor)

El modelo permite diseñar un bosquejo claro de las entidades, atributos y métodos con los que cuenta el sistema de anuncios publicitarios. La clase primordial del modelo es la tabla Anuncio, que tiene los atributos de fecha de publicación, fecha terminación y descripción del anuncio. Los métodos que se definen en cada una de las clases, son los siguientes: el método toString que permite devolver un texto al ser llamado el objeto de la clase, el método para realizar el registro de la información, el método que permite actualizar los datos registrados, que también permite deshabilitar elementos de cada una de las entidades y finalmente se tiene el método para obtener registros de la base de datos. La clase sucursal permite definir

las diferentes sedes que se registran de una empresa en cualquier lugar, para ofrecer su servicio; a la clase se define una relación obligatoria con las clases empresa, lugar y contacto. Finalmente se muestra la clase categoría, que permite la clasificación de las diferentes actividades económicas comerciales y que también facilitan la búsqueda de información en el sistema. Para cada clase se definieron atributos privados como la identificación, el nombre y el estado de los elementos.

5.4. Prototipos de la plataforma spot.co

Para el desarrollo del sistema de anuncios publicitarios, primero se diseñó la interfaz de usuario de la aplicación utilizando el software Balsamiq Mockups. En la figura 4 se puede observar el diseño de la página principal del usuario. Ver figura 4

Figura 4: Prototipo home (autor)

Para la administración de los anuncios publicitarios se diseñó el módulo general para realizar la gestión de los parámetros del sistema. En la figura 5 se puede observar el prototipo correspondiente al panel que tendrá la plataforma de spot.co para la administración de los módulos del sistema. Ver figura 5

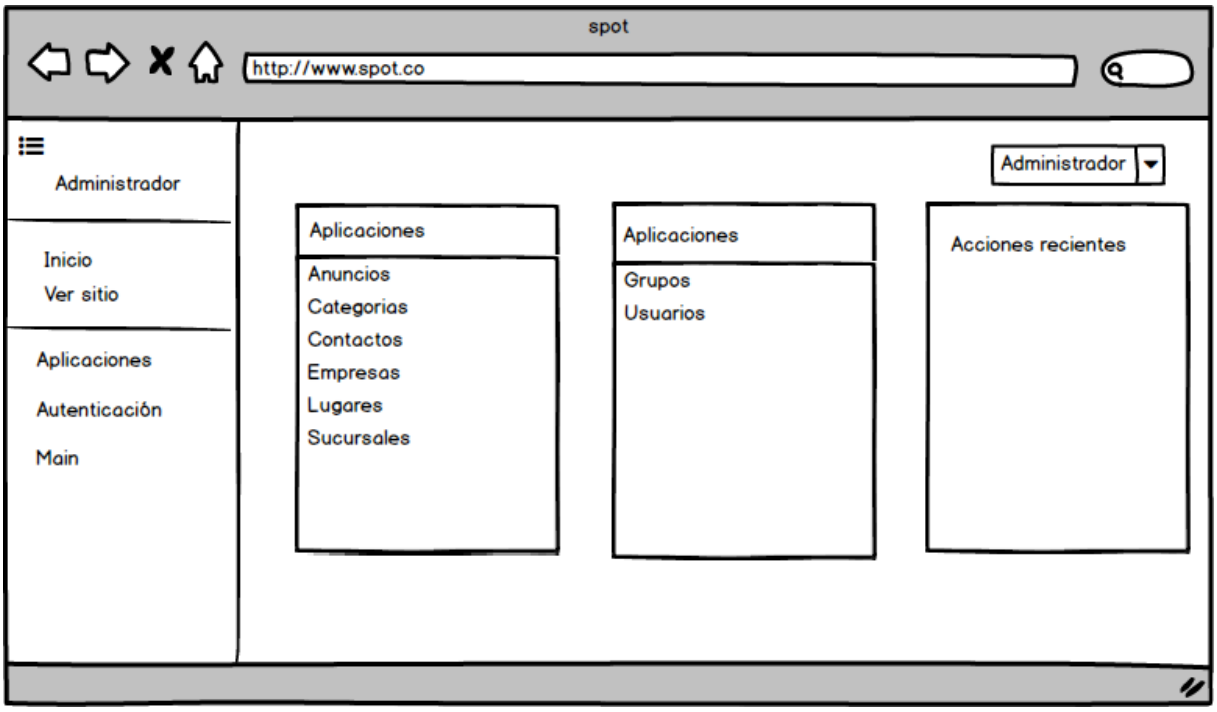


Figura 5: Prototipo administrador (autor)

El eje central del sistema **spot** son los anuncios publicitarios. Se diseñó las opciones que tendría dicho módulo para la gestión de la publicidad. En la figura 6 se puede observar la lista de anuncios por sucursal y la opción de agregar. Ver figura 6

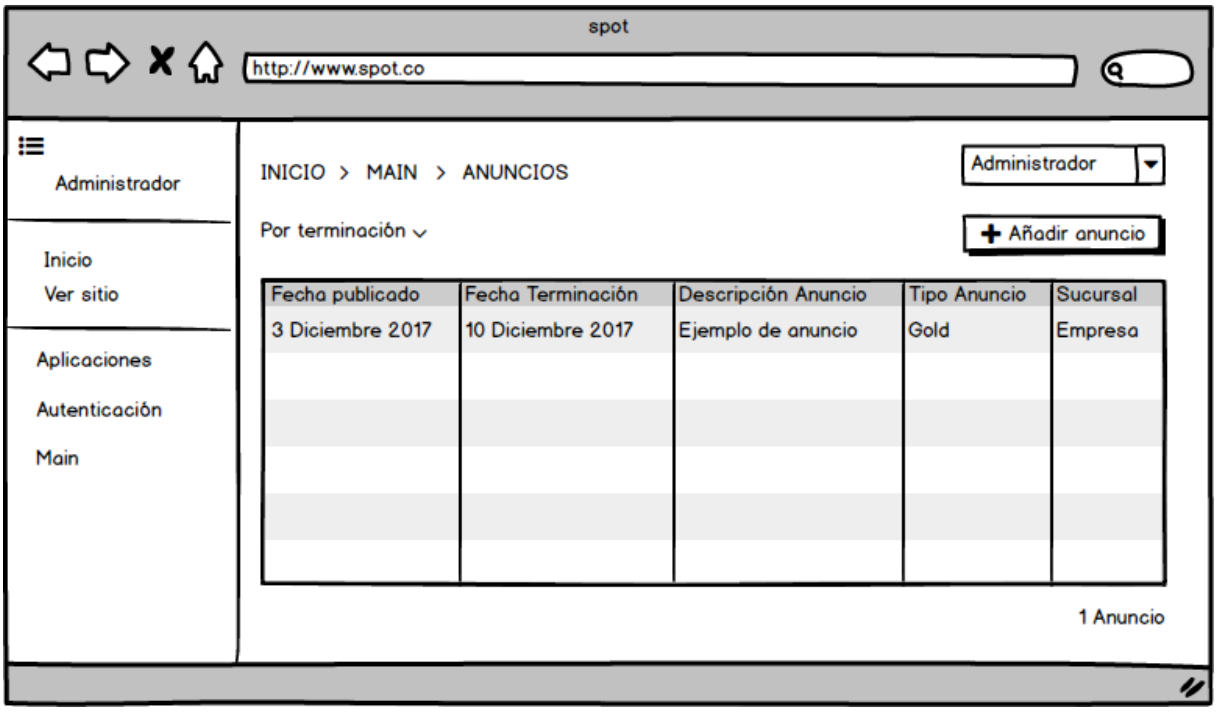


Figura 6: Lista de anuncios (autor)

En la figura 7 se puede observar el formulario para el registro de anuncios publicitarios. Cada formulario de registro tendrá la misma estructura. Ver figura 7

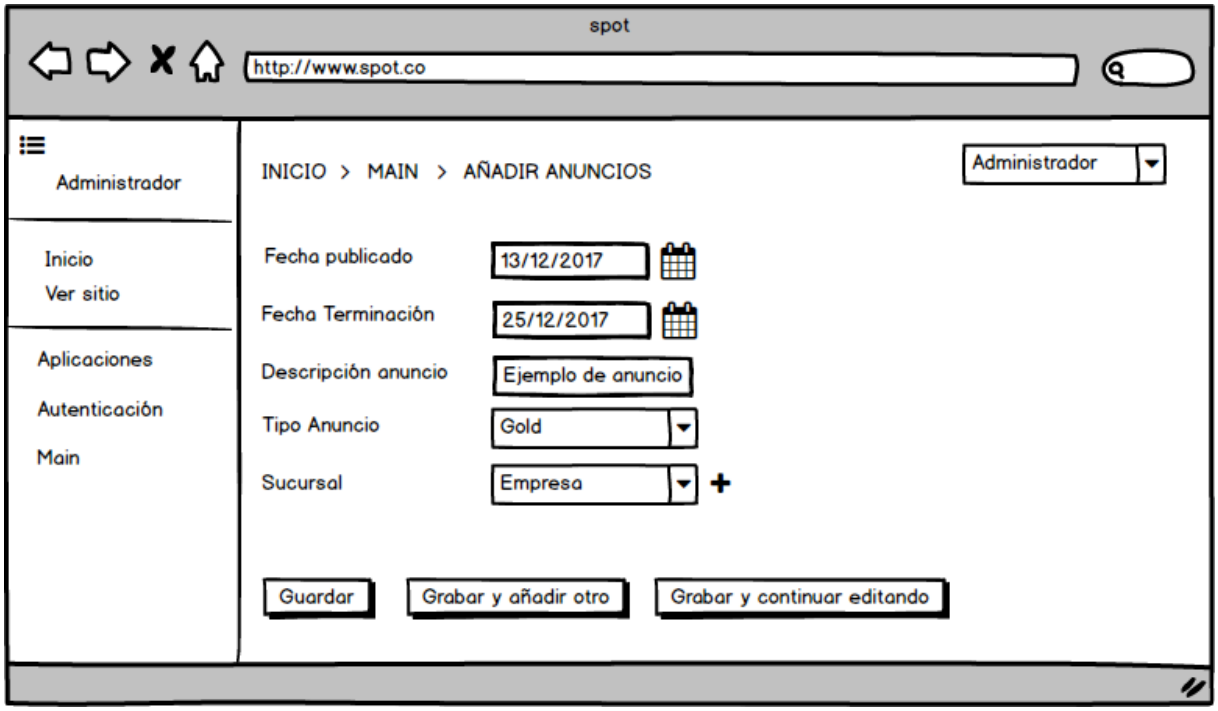


Figura 7: Registro de anuncios (autor)

El prototipo de la figura 8 muestra el módulo para la administración de categorías de las actividades económicas de las empresas. Ver figura 8

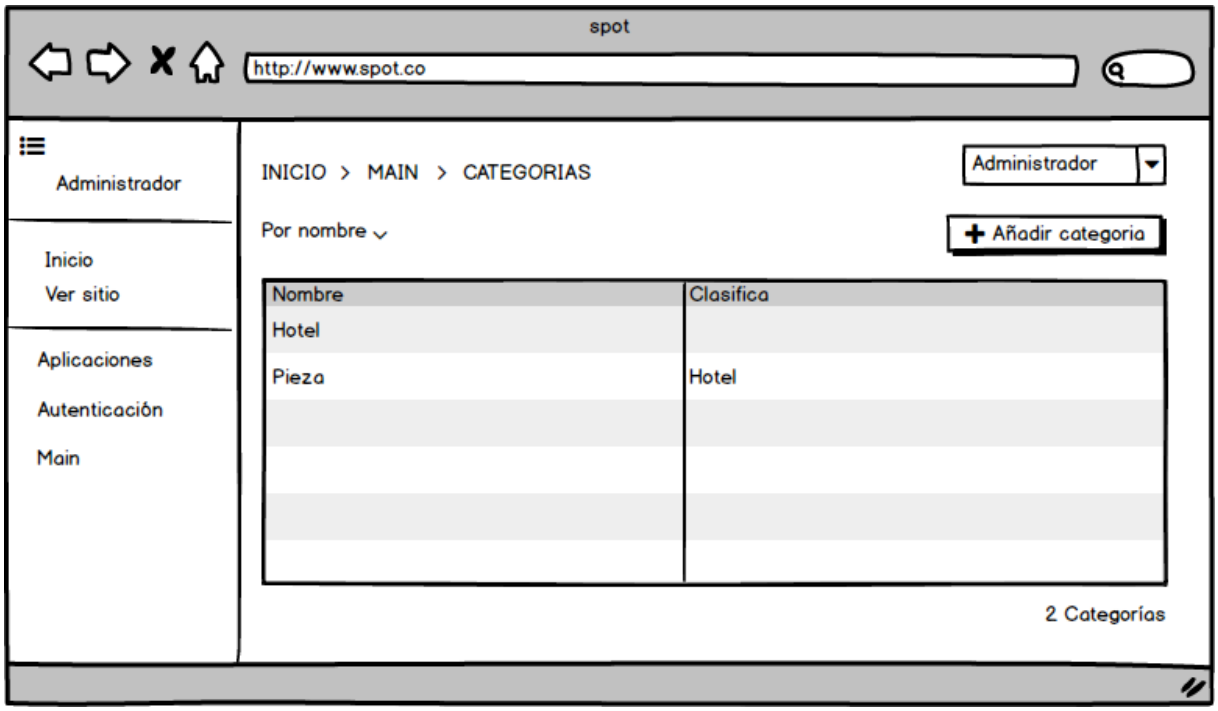


Figura 8: Prototipo para administrar categorías (autor)

El prototipo que se puede observar en la figura 9, permite realizar la administración de los contactos de cada empresa. Ver figura 9

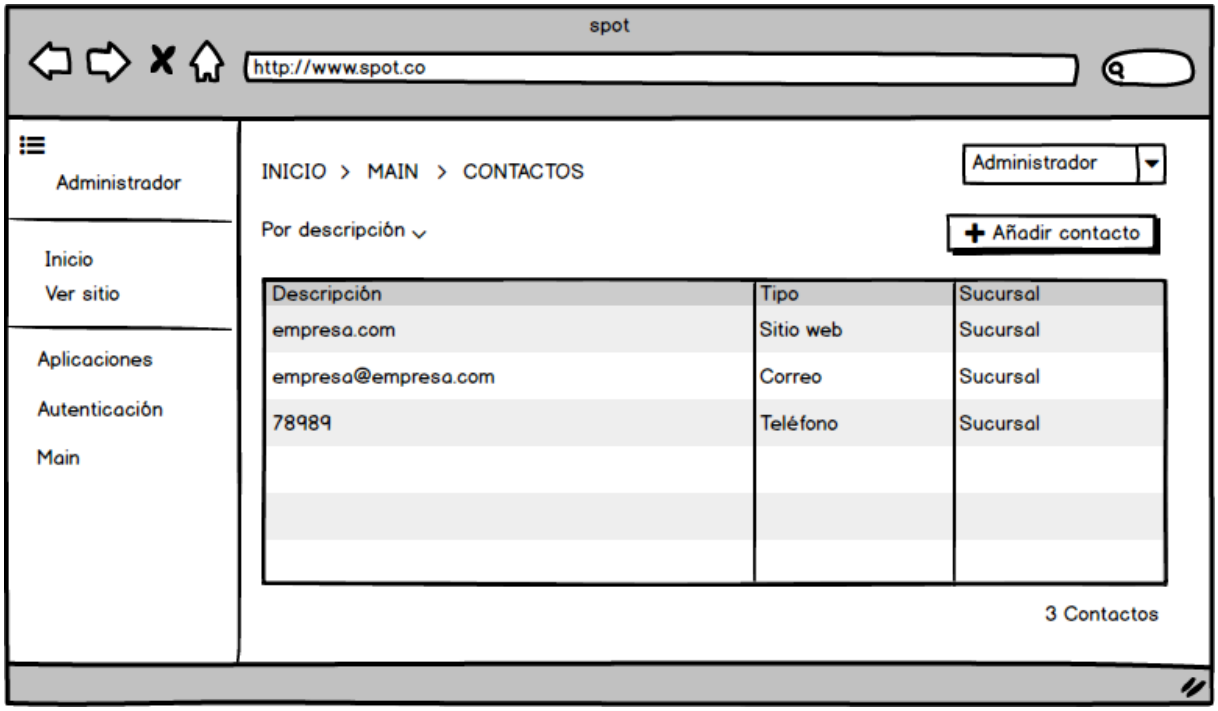


Figura 9: Prototipo administración de contactos (autor)

El prototipo de la figura 10 muestra el módulo para realizar la administración de las empresas por parte del usuario administrador. Ver figura 10

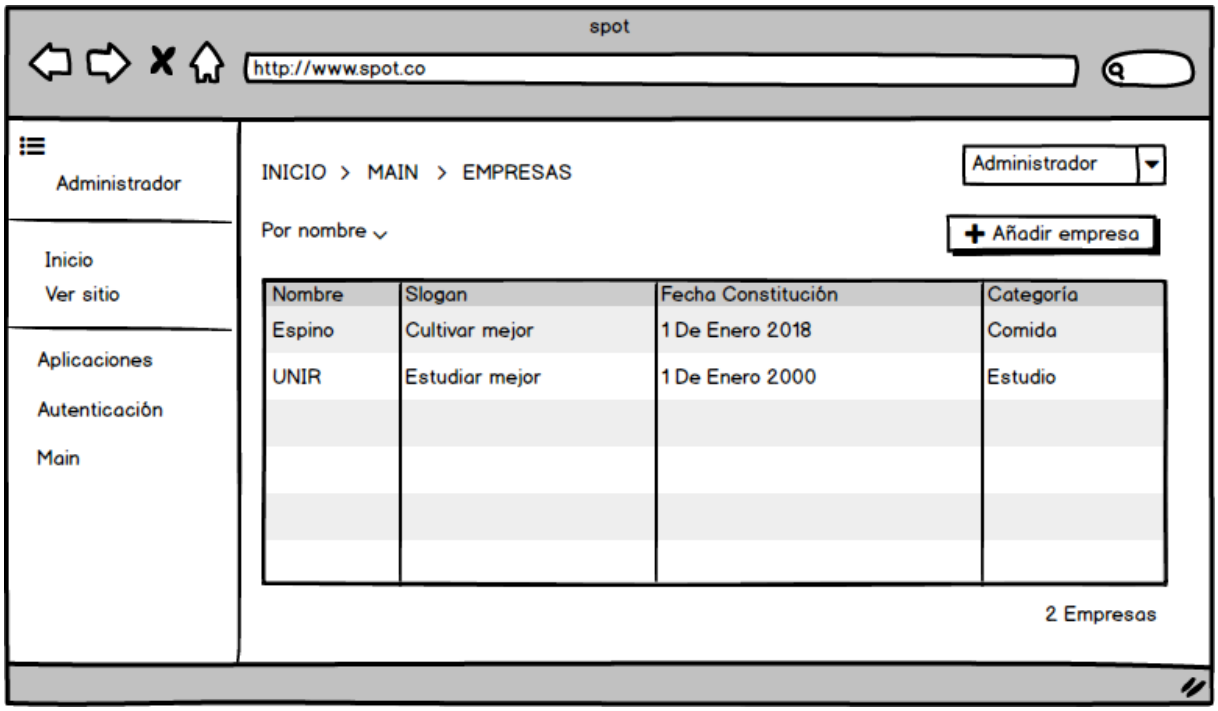


Figura 10: Prototipo módulo administración de empresas (autor)

El prototipo de la figura 11 muestra el módulo para realizar la administración los lugares en los que se registre la empresa. Ver figura 11

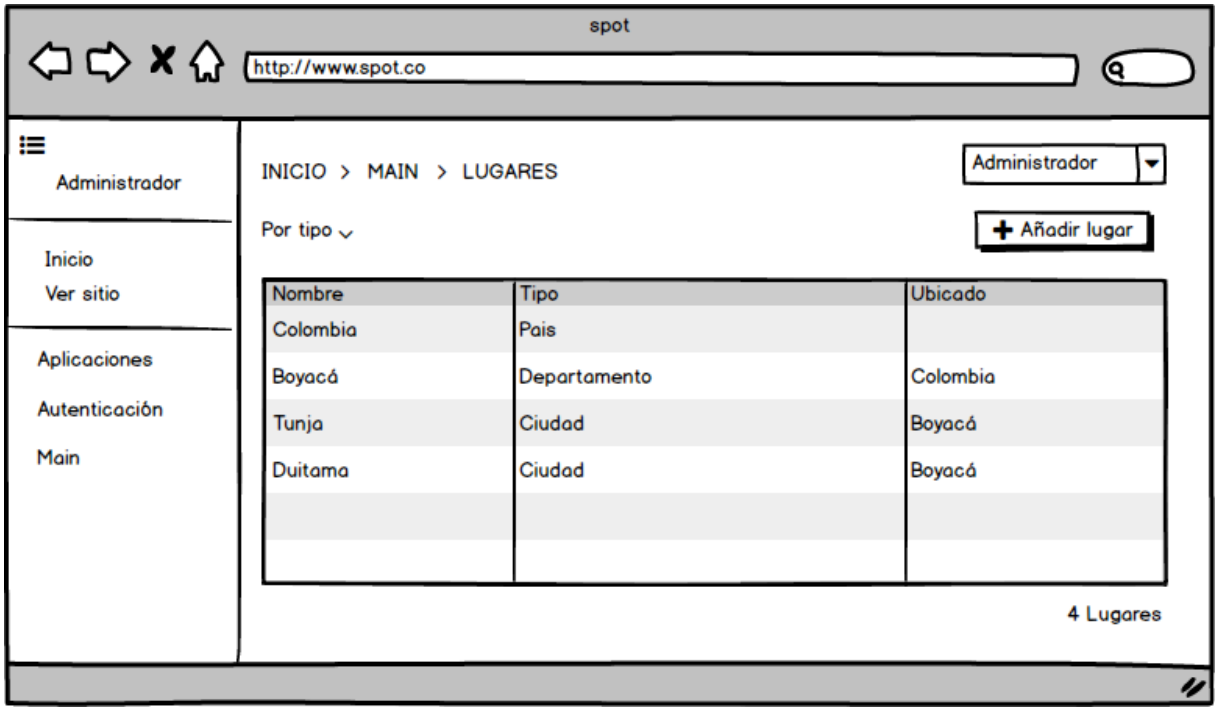


Figura 11: Prototipo administración lugares (autor)

El prototipo de la figura 12 muestra el módulo de administración de sucursales realizadas por parte del usuario administrador. Ver figura 12

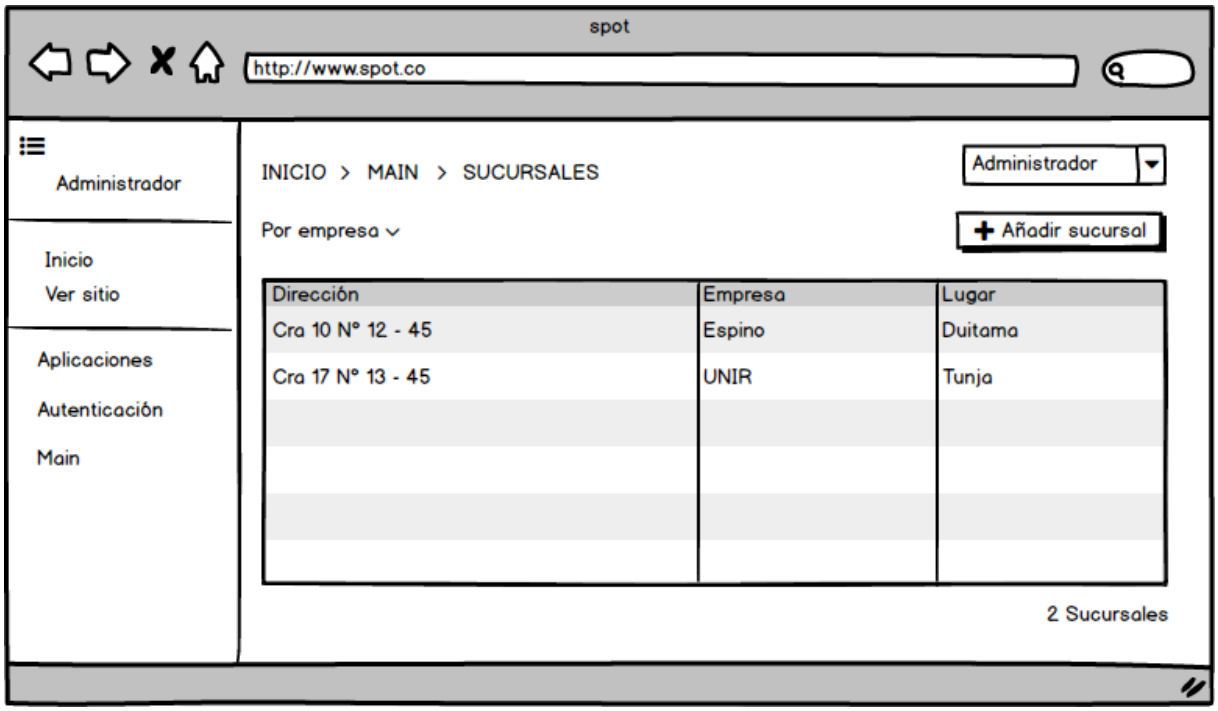


Figura 12: Prototipo administración de sucursales (autor)

5.5. Especificación del requisito en pruebas unitarias

Una vez definidas las historias de usuario y los diseños del software se procede a expresar dichas funciones en códigos de prueba. El Framework Django provee la API necesaria para realizar las pruebas unitarias de cada requisito del software. Cada prueba permite evaluar si un método dentro de un módulo funciona de forma correcta. Unittest que permite comprobar el comportamiento de los métodos de cada módulo. Para que el sistema detecte la prueba unitaria, cada método debe empezar con el nombre de *test*. [49] La tabla 26 muestra los diferentes métodos que dispone la herramienta para realizar las pruebas unitarias de cada uno de los requisitos del sistema. [50]

Tabla 26: Funciones pruebas unitarias

Función	Operación equivalente
assertEqual(a, b)	a == b
assertNotEqual(a, b)	a != b
assertTrue(x)	bool(x) is True
assertFalse(x)	bool(x) is False
assertIs(a, b)	a is b
assertIsNot(a, b)	a is not b
assertIsNone(x)	x is None
assertIsNotNone(x)	x is not None
assertIn(a, b)	a in b
assertNotIn(a, b)	a not in b
assertIsInstance(a, b)	isinstance(a, b)
assertNotIsInstance(a, b)	not isinstance(a, b)

Fuente: Adaptado de [50]

Para realizar las pruebas unitarias a las funciones del proyecto, se debe crear la clase de la entidad a analizar. En la figura 13 se puede observar las pruebas realizadas a la entidad de Anuncios. Para iniciar el proceso se debe utilizar el método **setUp** para inicializar objetos y preparar las variables a las que se les realizará el análisis. Para la prueba de la figura 13 se creó un elemento del tipo anuncio en la base de datos, para luego ser consultado a través de la llave primaria de dicho objeto y comparar la descripción del elemento con la registrada anteriormente en el método setUp y así verificar la veracidad de la información almacenada. Ver figura 13.

```python
from django.test import TestCase
from .models import *
'''
Autor           Oscar Iván Torres Corredor
Fecha           10/12/2017
'''
class AnuncioTestCase(TestCase):

    '''
    Autor           Oscar Iván Torres Corredor
    Fecha           10/12/2017
    Descripción     Inicializar modelos
    '''
    def setUp(self):
        categoria = Categoria.objects.create(nombre = "HOTELES")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion="Cra 1 N 10 - 50", empresa = empresa, lugar = lugar)
        Anuncio.objects.create(descripcion = "Ejemplo de anuncio publicitario para el proyecto.", sucursal= sucursal, categ

    '''
    Autor           Oscar Iván Torres Corredor
    Fecha           10/12/2017
    Descripción     Prueba unitaria: Obtener datos
    '''
    def testGetAnuncios(self):
        anuncio = Anuncio.objects.get(pk = 1)
        self.assertEqual(anuncio.descripcion, "Ejemplo de anuncio publicitario para el proyecto.")

    '''
```

Figura 13: Prueba obtener datos (autor)

De acuerdo al proceso de implementación del marco de trabajo TDD (Desarrollo de software orientado por pruebas) la prueba que se implemente en las fuentes del proyecto debe fallar como primer paso, para luego realizar la refactorización y corregir los inconvenientes presentados. Al implementar y ejecutar la prueba unitaria del paso anterior, en la figura 14 se puede observar que los textos comparados no son iguales y la herramienta que permite realizar el análisis lo puede detectar e indicarle al desarrollador la falla presentada para luego ser corregida. Ver figura 14

```
Símbolo del sistema
System check identified no issues (0 silenced).
.F................
======================================================================
FAIL: testGetAnuncios (main.tests.AnuncioTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\USER\Documents\advertising\main\tests.py", line 28, in testGetA
nuncios
    self.assertEqual(anuncio.descripcionAnuncio, "Ejemplo de anuncio publicitari
o para el proyecto SPOT.")
AssertionError: 'Ejemplo de anuncio publicitario para el proyecto.' != 'Ejemplo
de anuncio publicitario para el proyecto SPOT.'
- Ejemplo de anuncio publicitario para el proyecto.
+ Ejemplo de anuncio publicitario para el proyecto SPOT.
?                                                 +++++


----------------------------------------------------------------------
Ran 18 tests in 0.058s

FAILED (failures=1)
Destroying test database for alias 'default'...

(venv) C:\Users\USER\Documents\advertising>
```

Figura 14: Resultado prueba unitaria (autor)

Al realizar la refactorización de la prueba unitaria creada en la figura 13 y ejecutar dicha prueba de nuevo, utilizando el API del Framework de Django, se puede observar que el resultado de la figura 15; el sistema verifica la prueba y muestra un mensaje de exitoso. Ver figura 15

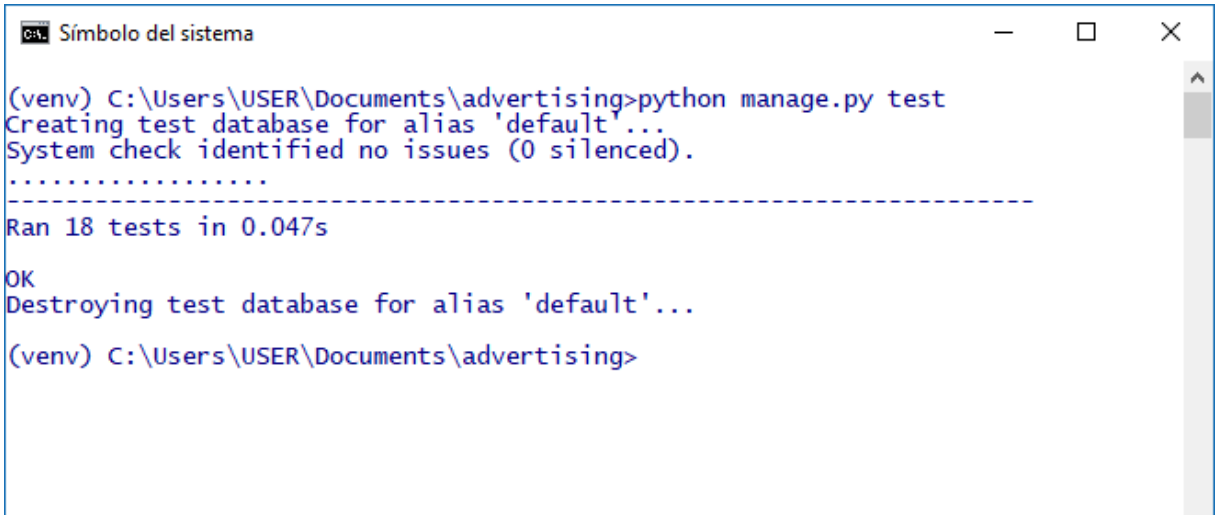


Figura 15: Resultado prueba unitaria (autor)

De acuerdo a las especificaciones del marco de trabajo TDD (Desarrollo orientado por pruebas) es conveniente que se realicen las suficientes pruebas unitarias, para comprender a la perfección los requisitos del sistema. En la figura 16 se pueden observar las pruebas unitarias para los requisitos de obtener listado de datos y registrar información de la entidad de Anuncios. Ver figura 16

```
    '''
    Autor           Oscar Iván Torres Corredor
    Fecha           10/12/2017
    Descripción     Prueba unitaria: Contar lista
    '''
    def testCountAnuncios(self):
        anuncio = Anuncio.objects.all();
        self.assertEqual(anuncio.count(), 1)

    '''
    Autor           Oscar Iván Torres Corredor
    Fecha           10/12/2017
    Descripción     Prueba unitaria: Guardar datos
    '''
    def testSaveAnuncios(self):
        categoria = Categoria.objects.create(nombre = "TAXIS")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion = "Cra 1 N 10 - 50", empresa = empresa, lugar = lugar)
        anuncio = Anuncio.objects.create(descripcion = "Ejemplo de anuncio publicitario para el proyecto.", sucursal= sucur
        self.assertTrue(anuncio)
```

Figura 16: Prueba unitaria listar y guardar (autor)

Es conveniente verificar que todas las pruebas unitarias creadas para el sistema resulten correctas. En la figura 17 se puede observar que al ejecutar el comando para verificar cada prueba, la consola arroja un mensaje del correcto funcionamiento. Ver figura 17

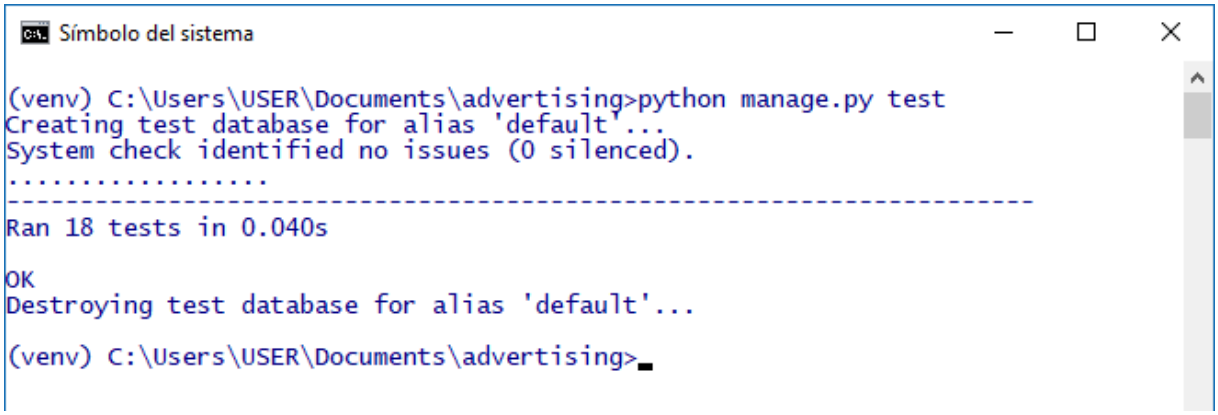


Figura 17: Resultados pruebas unitarias (autor)

En la figura 18 se muestran las pruebas unitarias realizadas para la entidad de categorías. Es importante realizar las suficientes pruebas, que permitan comprender de la mejor manera el contexto para el que fue creada la entidad categorías. Ver figura 18

```
class CategoriaTestCase(TestCase):
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Inicializar variables'''
    def setUp(self):
        Categoria.objects.create(nombre = "RESTAURANTES")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Obtener dato'''
    def testGetCategoria(self):
        categoria = Categoria.objects.get(pk = 1)
        self.assertEqual(categoria.nombre, "RESTAURANTES")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Contar registros'''
    def testCountCategorias(self):
        categoria = Categoria.objects.all()
        self.assertEqual(categoria.count(), 1)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Guardar datos'''
    def testSaveCategorias(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        self.assertTrue(categoria)
```

Figura 18: Test Unitario Categorías (autor)

En la figura 19 se muestran las pruebas unitarias realizadas para la entidad de Lugares. De acuerdo a las historias planteadas para el proyecto, se deben realizar pruebas unitarias para los requisitos de crear, actualizar, ver y deshabilitar un elemento de la entidad. Ver figura 19

```python
class LugarTestCase(TestCase):
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Inicializar datos'''
    def setUp(self):
        pais = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        departamento = Lugar.objects.create(nombre = "Boyacá", tipo = "D", ubicadoEn = pais)
        ciudad = Lugar.objects.create(nombre = "Tunja", tipo = "P", ubicadoEn = departamento)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Obtener datos'''
    def testGetLugar(self):
        pais = Lugar.objects.get(pk = 1)
        departamento = Lugar.objects.get(pk = 2)
        ciudad = Lugar.objects.get(pk = 3)
        self.assertEqual(pais.nombre, "Colombia")
        self.assertEqual(departamento.nombre, "Boyacá")
        self.assertEqual(ciudad.nombre, "Tunja")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Lista datos'''
    def testCountLugares(self):
        lugares = Lugar.objects.all()
        self.assertEqual(lugares.count(), 3)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Guardar datos'''
    def testSaveLugar(self):
        pais = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        departamento = Lugar.objects.create(nombre = "Boyacá", tipo = "D", ubicadoEn = pais)
        ciudad = Lugar.objects.create(nombre = "Tunja", tipo = "P", ubicadoEn = departamento)
        self.assertTrue(ciudad)
```

Figura 19: Test Unitario Lugares (autor)

En la figura 20 se muestran las pruebas unitarias realizadas para la entidad de Empresas. Se crearon las pruebas unitarias para obtener, listar y crear datos de las empresas. Ver figura 20

```python
class EmpresaTestCase(TestCase):
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Inicializar datos'''
    def setUp(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        Empresa.objects.create(nombre = "Espino", slogan = "Lo mejor para ti", categoria = categoria)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Obtener datos'''
    def testGetEmpresa(self):
        empresa = Empresa.objects.get(pk = 1)
        self.assertEqual(empresa.nombre, "Espino")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Lista datos'''
    def testCountEmpresa(self):
        empresas = Empresa.objects.all()
        self.assertEqual(empresas.count(), 1)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Guardar datos'''
    def testSaveEmpresa(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        empresa = Empresa.objects.create(nombre = "Espino", slogan = "Lo mejor para ti", categoria = categoria)
        self.assertTrue(empresa)
```

Figura 20: Test Unitario Empresas (autor)

En la figura 21 se muestran las pruebas unitarias realizadas para la entidad Sucursal. Al igual que las demás entidades, se realizaron las pruebas necesarias para verificar y comprender el contexto de las sucursales de la plataforma web. Ver figura 21

```python
class SucursalTestCase(TestCase):
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Inicializar datos'''
    def setUp(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion="Cra 1 N 10 - 20", empresa = empresa, lugar = lugar)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Obtener datos'''
    def testGetSucursal(self):
        sucursal = Sucursal.objects.get(direccion = "Cra 1 N 10 - 20")
        self.assertEqual(sucursal.direccion, "Cra 1 N 10 - 20")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Lista datos'''
    def testCountSucursal(self):
        sucursales = Sucursal.objects.all()
        self.assertEqual(sucursales.count(), 1)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Guardar datos'''
    def testSaveSucursal(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion ="Cra 1 N 10 - 20", empresa = empresa, lugar = lugar)
        self.assertTrue(sucursal)
```

Figura 21: Test Unitario Sucursales (autor)

En la figura 22 se muestran las pruebas unitarias realizadas para la entidad de Contactos. De acuerdo a la documentación, es recomendable realizar la cantidad de pruebas unitarias necesarias, que permitan comenzar el desarrollo del proyecto de forma objetiva; para el presente proyecto se tuvo en cuenta la identificación de requisitos y se crearon las pruebas necesarias para entender el contexto general del funcionamiento de la aplicación. Ver figura 22

```python
class ContactoTestCase(TestCase):
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Inicializar datos'''
    def setUp(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion ="Cra 1 N 10 - 20", empresa = empresa, lugar = lugar)
        Contacto.objects.create(descripcion = "Ejemplo de Contacto", tipo = "W", sucursal=sucursal)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Obtener datos'''
    def testGetContacto(self):
        contacto = Contacto.objects.get(pk = 1)
        self.assertEqual(contacto.descripcion, "Ejemplo de Contacto")
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Listar datos'''
    def testCountContacto(self):
        contactos = Contacto.objects.all()
        self.assertEqual(contactos.count(), 1)
    ''' Autor           Oscar Iván Torres Corredor
        Descripción     Prueba unitaria: Guardar datos'''
    def testSaveContacto(self):
        categoria = Categoria.objects.create(nombre = "DROGUERIA")
        lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
        empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
        sucursal = Sucursal.objects.create(direccion = "Cra 1 N 10 - 20", empresa = empresa, lugar = lugar)
        contacto = Contacto.objects.create(descripcion = "Ejemplo de Contacto", tipo = "W", sucursal=sucursal)
        self.assertTrue(contacto)
```

Figura 22: Test Unitario Contactos (autor)

El Framework Django y la API de Unit Test, permiten verificar el correcto funcionamiento de todas las pruebas unitarias; para esto se debe ejecutar el comando que se puede ver en la figura 23 y que a continuación se describe: Python manage.py test. Ver figura 23

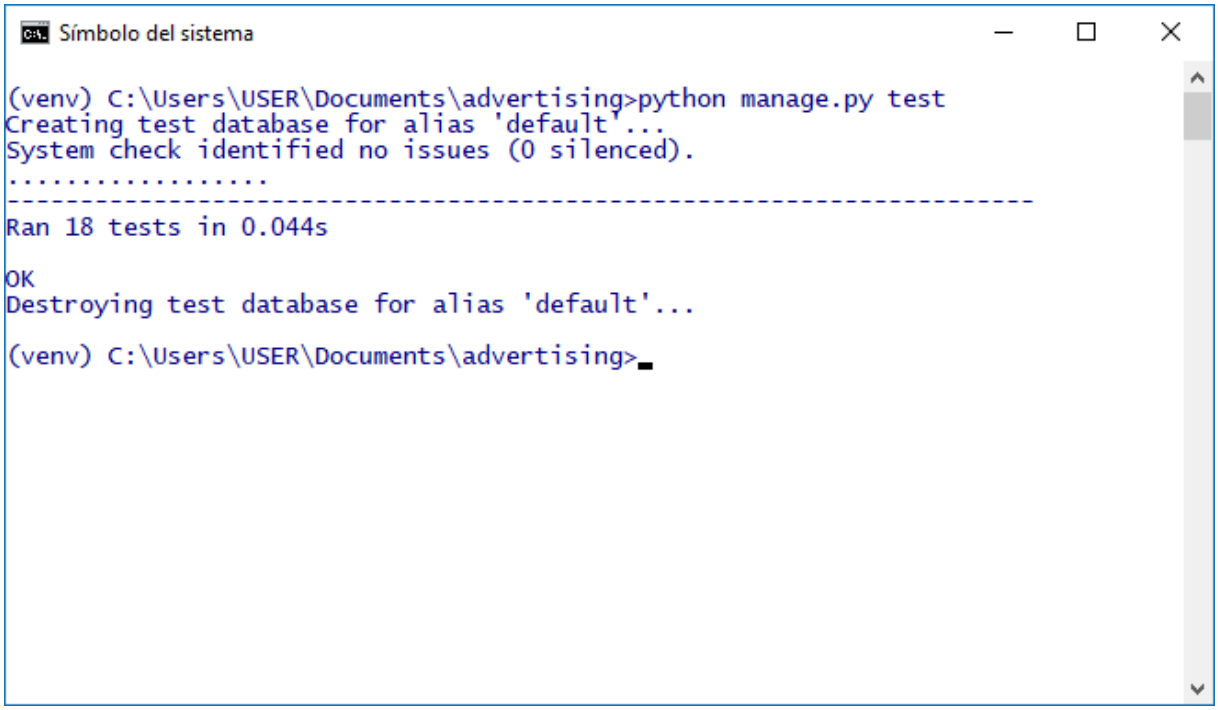


Figura 23: Verificar Test Unitario (autor)

En la figura 24 se muestra el resultado en el que se traduce cada una de las pruebas unitarias. Las pruebas unitarias tan solo es un modelo que permite comprender ampliamente las funcionalidades del sistema y comenzar la codificación inicial del proyecto; pero al momento de pasar a un ambiente de desarrollo final, las fuentes y métodos son sometidos a grandes cambios, teniendo en cuenta que se deben obtener datos desde los formularios, realizar validaciones de los campos en el lado del servidor y cambios de formato para algunos datos; por ejemplo en la figura 24, para registrar las fechas del formulario de anuncios, se debe realizar una transformación del formato, que viene desde el formulario de registro de anuncios, para que el sistema lo pueda ingresar a la base de datos.

```
''' Autor           Oscar Iván Torres Corredor
    Descripción     Listar datos
    '''
@login_required(login_url='login/')
def listarAnuncios(request):
    anuncios = Anuncio.objects.all()
    return render( request ,'anuncio/lista.html', locals())
''' Autor           Oscar Iván Torres Corredor
    Descripción     Deshabilitar datos
    '''
@login_required(login_url='login/')
def deshabilitarAnuncio(request):
    if request.method == 'POST':
        idAnuncio = request.POST.get('pk', None)
        anuncio = Anuncio.objects.get(pk = idAnuncio)
        anuncio.estado = 0
        anuncio.save()
        return HttpResponseRedirect('/anuncio/lista/')
''' Autor           Oscar Iván Torres Corredor
    Descripción     Guardar datos
    '''
@login_required(login_url='login/')
def guardarAnuncio(request):
    if request.POST:
        publicado = request.POST.get('publicado', None)
        terminacion = request.POST.get('terminacion', None)
        tipo = request.POST.get('tipo', None)
        descripcion = request.POST.get('descripcion', None)
        sucursal = request.POST.get('sucursal', None)
        categoria = request.POST.get('categoria', None)
        sucursale = Sucursal.objects.get(pk =  sucursal)
        category = Categoria.objects.get(pk = categoria)
        anuncio = Anuncio()
        anuncio.publicado = datetime.datetime.strptime(publicado,"%m/%d/%Y").strftime("%Y-%m-%d")
        anuncio.publicado = datetime.datetime.strptime(terminacion,"%m/%d/%Y").strftime("%Y-%m-%d")
        anuncio.tipo = tipo
        anuncio.descripcion = descripcion
        anuncio.sucursal = sucursale
        anuncio.categoria = category
        anuncio.save()
    return HttpResponseRedirect('/anuncio/lista/' )
```

Figura 24: Funcionalidades del software (autor)

Al finalizar la implementación de la metodología de desarrollo de software orientado por pruebas, es importante señalar que para efectos del presente documento solo se muestran algunos ejemplos de pruebas unitarias, para identificar el proceso que tiene el marco de trabajo TDD; pero a nivel de proyecto hay más pruebas realizadas, que no pueden ser agregadas al documento, para así cumplir de esta manera con los requisitos mínimos que exige la Universidad y no sobre pasar el límite permitido.

5.6. Plataforma de anuncios publicitarios

Al finalizar las pruebas unitarias y entender ampliamente las funcionalidades de la plataforma de anuncios publicitarios, se procede a codificar el software. La metodología TDD precisa en sus procesos, que al contar con las pruebas unitarias totalmente correctas y organizadas, dichos códigos son traslados como funcionalidades de las fuentes finales del proyecto. En la

figura 25 se puede observar la página principal del portal de anuncios publicitarios. Se puede observar la importancia del buscador, que es el objetivo principal del portal. Ver figura 25

Figura 25: *Portal spot.co (autor)*

En la figura 26 se muestra la lista de anuncios registrados en el sistema, al ser realizada la consulta en el portal principal del sistema de publicidad. Ver figura 26

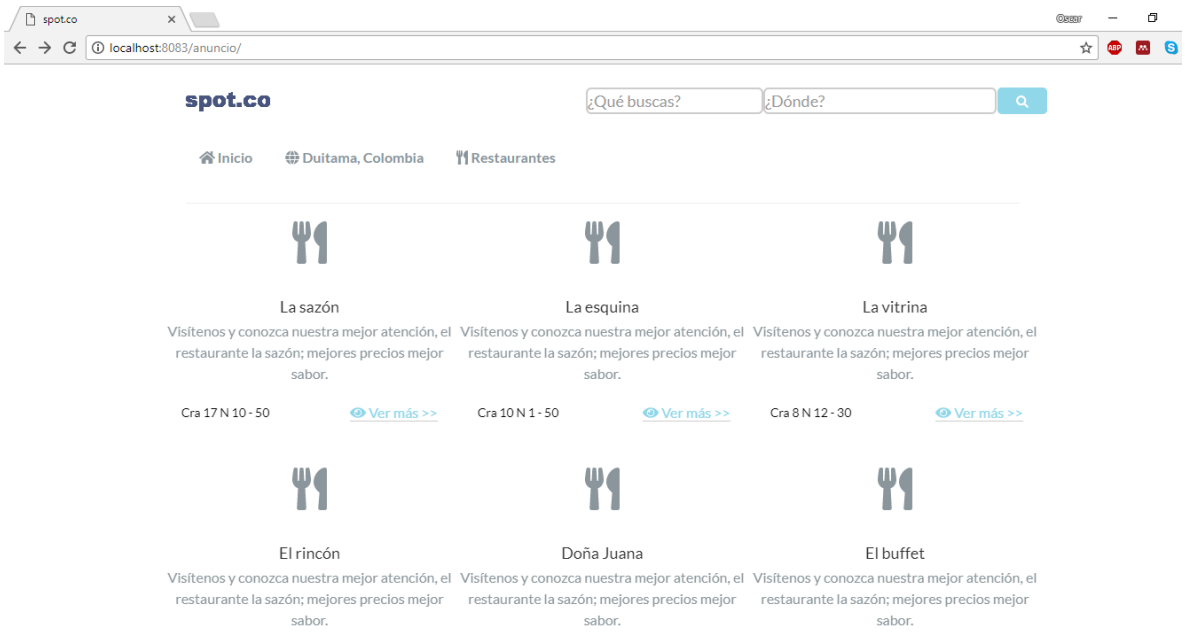


Figura 26: Lista anuncios (autor)

El objetivo primordial del sitio web es la administración de anuncios publicitarios y su visualización a los usuarios del sistema, por lo tanto se desarrolló un módulo administrador que permitiera la gestión de dicha publicidad en el sistema. La figura 27 permite visualizar el módulo para la administración de anuncios publicitarios por parte del usuario administrador. Ver figura 27

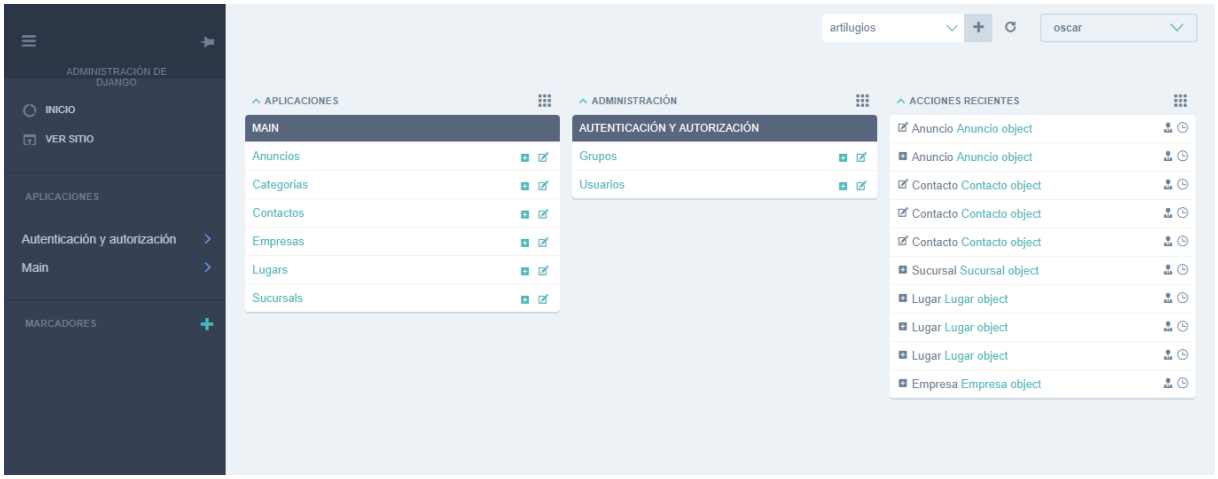


Figura 27: Módulo administrador (autor)

En la figura 28 se puede observar el módulo para la gestión de los anuncios publicitarios por parte del usuario administrador. Ver figura 28

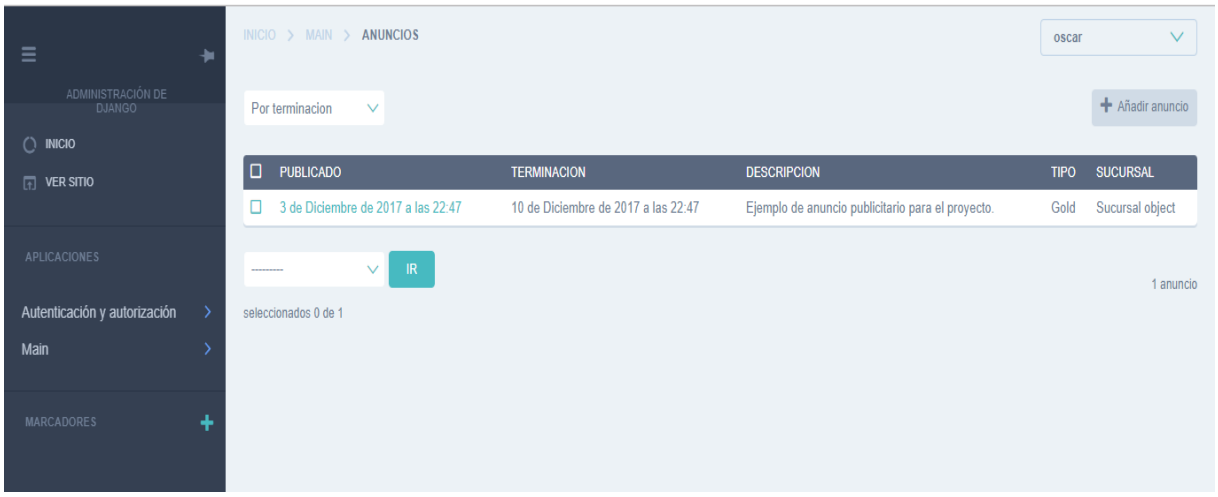


Figura 28: Listado de anuncios publicitarios (autor)

En la figura 29 se puede observar el formulario para el registro de anuncios publicitarios en la plataforma de spot.co, por parte del usuario administrador. Ver figura 29

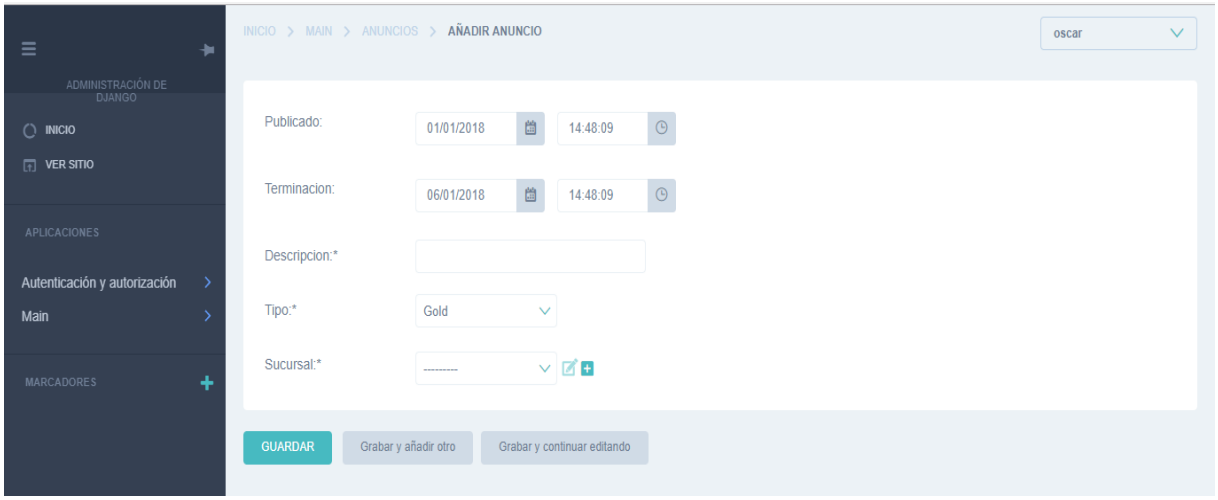


Figura 29: Registrar anuncios publicitarios (autor)

En la figura 30 se muestra el formulario que permite realizar la edición de los datos por parte del usuario administrador. También permite eliminar el elemento seleccionado. Ver figura 30

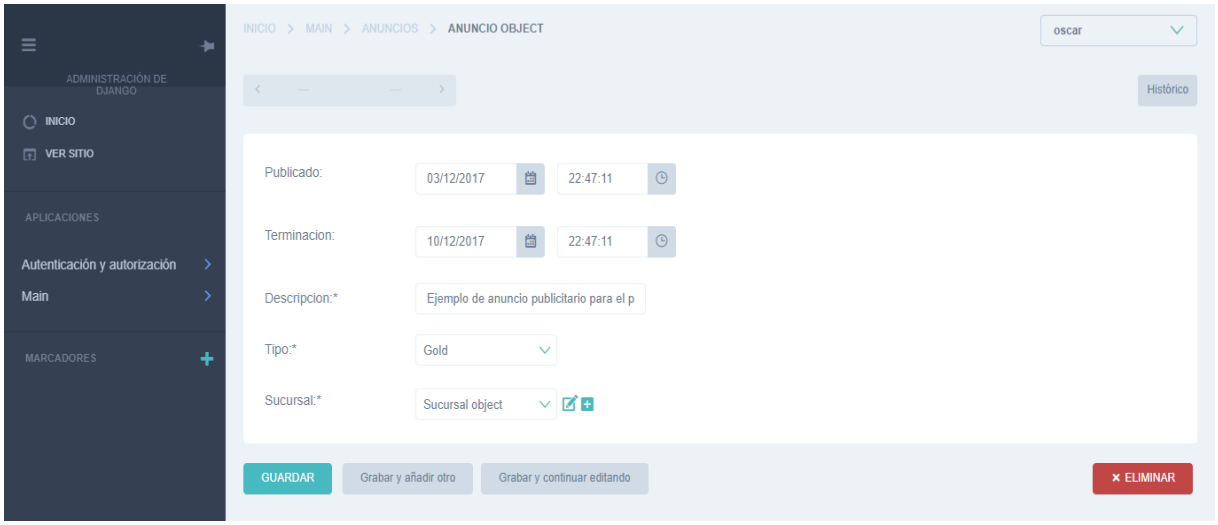


Figura 30: Editar anuncios (autor)

En la figura 31 se muestra el módulo para la gestión de categorías por parte del usuario administrador. Ver figura 31

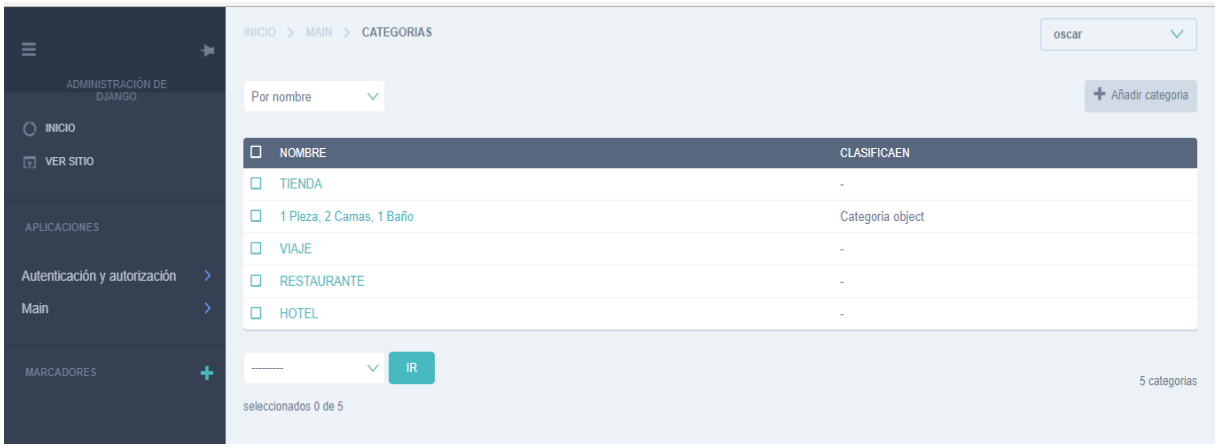


Figura 31: Administrar categorías (autor)

En la figura 32 se puede observar el módulo para la administración de contactos de cada sucursal, realizados por el usuario administrador. Ver figura 32

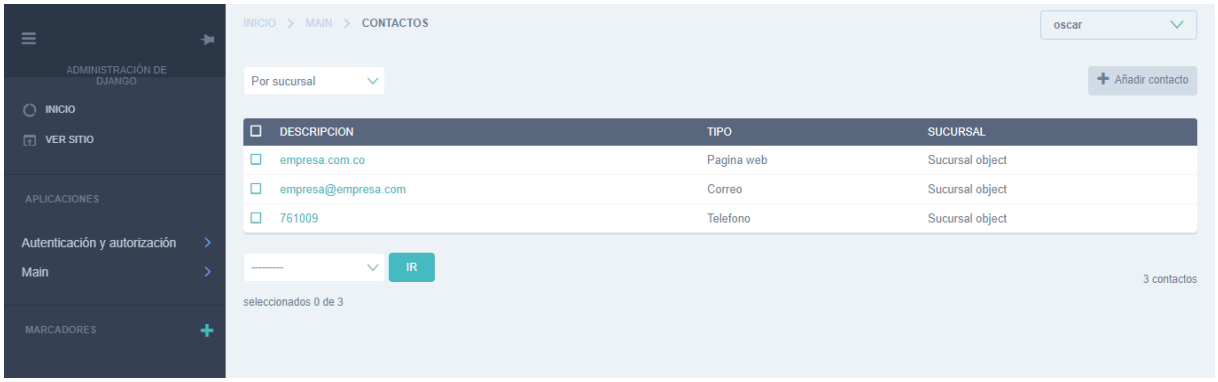


Figura 32: Administrar contactos (autor)

En la figura 33 se puede observar el módulo para la administración de empresas, realizados por el usuario administrador. Ver figura 33

Figura 33: Administración de empresas (autor)

En la figura 34 se puede observar el módulo para la administración de los lugares de las sucursales de las empresas, realizados por el usuario administrador. Ver figura 34

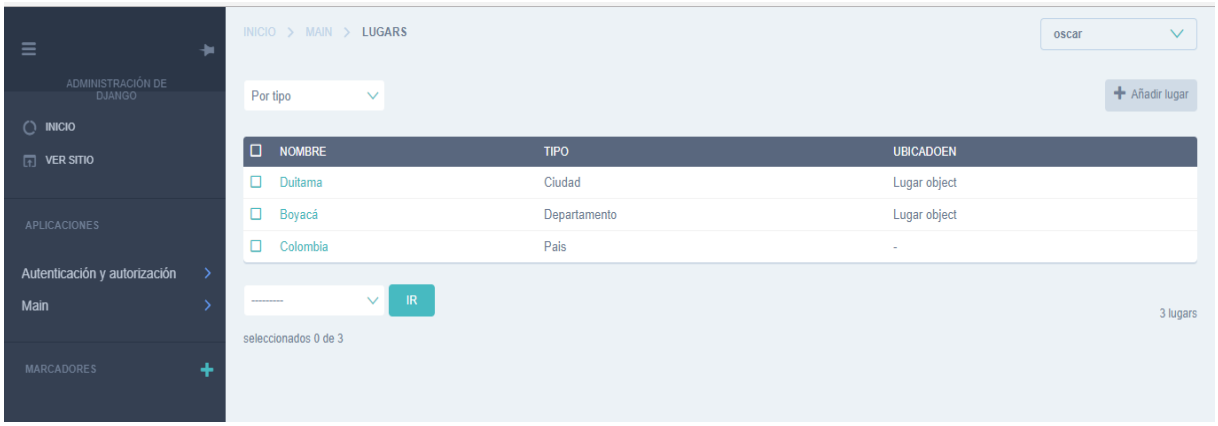


Figura 34: Administración lugares (autor)

En la figura 35 se puede observar el módulo para la administración de sucursales de las empresas, realizados por el usuario administrador. Ver figura 35

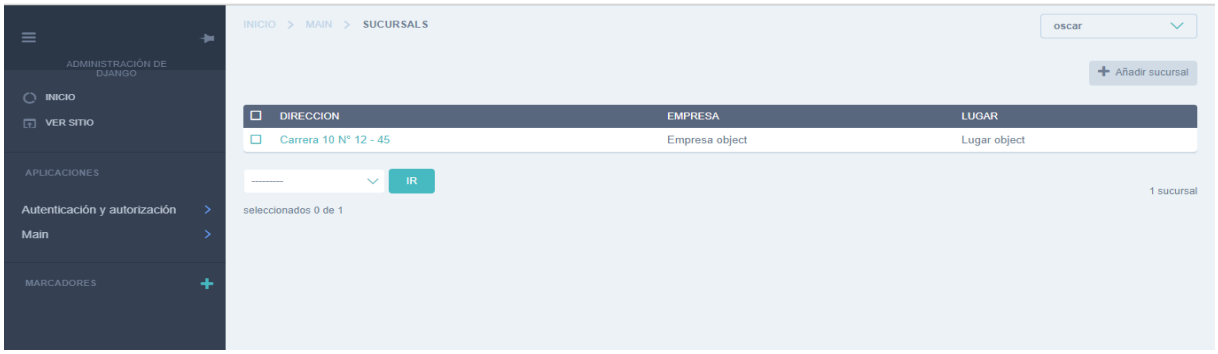


Figura 35: Administración de sucursales (autor)

5.7. Modelo de calidad ISO/IEC 9126

De acuerdo a los objetivos del presente proyecto, una de las partes importantes es la apreciación de la calidad del producto de software. Para realizar la evaluación de calidad del producto de software del proyecto de anuncios publicitarios, se tuvo en cuenta el estándar de la ISO/IEC 9126. A continuación se describe el proceso:

5.7.1. Atributos del modelo de calidad de la ISO/ IEC 9126

Para evaluar la aptitud del producto software de este proyecto se tuvo en cuenta las características propuestas por el modelo de la ISO/IEC 9126. Para cada característica se describe de forma general los procesos, técnicas y/o herramientas empleadas para dar cumplimiento a las recomendaciones descritas en el modelo de calidad. A continuación se presentan los factores evaluados.

5.7.1.1 Funcionalidad

Mediante una correcta apropiación las funcionalidades que debe tener un software de anuncios publicitarios, se definieron los requisitos que satisfacen las necesidades del proyecto, así al final del proyecto se obtuvo un producto software que cumple con las expectativas y exigencias de una plataforma de publicidad. Este sistema de información administra los datos de los anuncios publicitarios y los publica para ser consultados por la sociedad en general.

5.7.1.2. Usabilidad

El sistema de información brinda un entorno agradable, familiar y ameno a los usuarios permitiéndoles desarrollar sus actividades de forma efectiva y eficiente. La utilización de diseños dinámicos de interfaz, el uso de iconos, los formularios cortos y la implementación de ventanas emergentes contribuyeron a hacer ameno el uso del sistema, logrando que el costo de aprendizaje de la estructura lógica del software se pueda realizar lo más rápido posible.

5.7.1.3. Eficiencia

El uso de la tecnología del Framework de Django y el módulo de administración, permite desarrollar plataformas livianas y con tiempos de respuesta mínimos, alcanzando mayor grado de interactividad, velocidad y usabilidad en la aplicación, además con la utilización de

JavaScript en los clientes para validar los formularios no se satura el servidor de peticiones, contribuyendo así a obtener tiempos de respuesta cortos.

5.7.1.4. Mantenibilidad

Las variables y procesos involucrados en el uso de la metodología TDD (Desarrollo orientado por pruebas) y la plataforma de anuncios publicitarios y las diferentes funcionalidades que se le pueden agregar al software en el tiempo, pueden variar; para satisfacer la demanda de nuevas características al sistema, se diseñó un modelo de base de datos que se ajuste dinámicamente a posibles cambios en el manejo de información. Además aprovechando las ventajas de la arquitectura implementada (MVC) se divide la aplicación en tres componentes lo que facilita las tareas de actualización o corrección de fallos.

5.7.1.5. Portabilidad

Las aplicaciones web necesitan únicamente un navegador de internet para ser ejecutadas, por tal razón se garantiza portabilidad del sistema desarrollado en los navegadores Mozilla Firefox 36.0.34, Google Chrome 42.0.2311 y Opera 28.0.1750, o en versiones superiores.

5.7.2. Características ISO/IEC 9126 para evaluar el software

A continuación se describen las características que se tuvieron en cuenta para realizar el análisis e implementación de la norma de la ISO/IEC 9126 al sistema desarrollado teniendo en cuenta los procesos del marco de trabajo del desarrollo orientado por pruebas (TDD). Para la implementación se tuvo en cuenta la investigación [16] para determinar la forma de realizar el análisis del software de acuerdo a lo establecido por la norma.

Tabla 27: Características ISO/IEC 9126

ISO/IEC 9126		CARACTERÍSTICAS
Funcionalidad	Idoneidad	I1: Nombre de etiquetas
		I2: Nombre de títulos
	Exactitud	E1: Exactitud de las funciones
	Interoperabilidad	P1: Interacción con otro sistema
	Seguridad	S1: Inicio de sesión
		S2: Control de acceso
		S3: Redirección a sitio autorizado
Fiabilidad	Tolerancia a fallos	T1: Manejo de errores

Usabilidad	Facilidad de comprensión	F1: Facilidad de acceso al buscador
		F3: Uso de colores
	Facilidad de aprendizaje	A1:Mensajes de ayuda
		A2: Multimedia
		A3: Imágenes
		A4: Iconos y descripciones
	Operatividad	O1: Esfuerzo
Eficiencia	Tiempo de uso	R1: Tiempo de respuesta
	Recursos utilizados	U1: Recursos utilizados
Mantenibilidad	Simplicidad de análisis	D1: Documentación fuentes
	Simplicidad de cambio	C1: Registro de cambios
	Estabilidad	M1: Estabilidad del sistema
	Facilidad de prueba	G1: Facilidad de pruebas de aceptación
Portabilidad	Facilidad de instalación	L1: Multiplataforma
	Facilidad de adaptación al cambio	H1: Adaptación al cambio

Fuente: autor

5.7.3. Resultado implementación norma ISO/IEC 9126

A continuación se muestra el resultado del análisis e implementación de la norma ISO/IEC 9126 al software desarrollado bajo el marco de trabajo desarrollo orientado por pruebas. La tabla 28 muestra los pesos que se le asignaron a cada característica del modelo de calidad, partiendo de la experiencia que tuvo el usuario al operar el sistema y responder la encuesta que se encuentra en los anexos del proyecto. Es importante señalar que las preguntas realizadas en la encuesta de satisfacción por parte del usuario se tuvieron en cuenta los atributos de usabilidad y funcionalidad del estándar de calidad de la ISO/IEC 9126, debido a que es un aspecto que lo puede evaluar un usuario final, mientras que las demás características fue necesario realizar la consulta a un experto en el tema para poder asignar una valoración del nivel de calidad de las características. De acuerdo a la metodología consultada [16], la evaluación se realiza de manera cualitativa, sin embargo para facilidad del proceso, se maneja una escala de 0 a 1, repartiendo las valoraciones de la siguiente manera:

Bajo (B): 0,00 a 0,33

Medio (M): 0,34 a 0,75

Alto (A): 0,76 a 1,00

Tabla 28: Resultados implementación estándar ISO/IEC 9126

CARACTERÍSTICAS	PUNTAJE	TOTAL CARACTERÍSTICA
I1: Nombre de etiquetas	0,95	0,935
I2: Nombre de títulos	0,92	
E1: Exactitud de las funciones	1	1
P1: Interacción con otro sistema	1	1
S1: Inicio de sesión	1	1
S2: Control de acceso	1	
S3: Redirección a sitio autorizado	1	
T1: Manejo de errores	0,94	0,94
F1: Facilidad de acceso al buscador	1	0,975
F3: Uso de colores	0,95	
A1:Mensajes de ayuda	0,96	0,94
A2: Multimedia	0,92	
A3: Imágenes	0,93	
A4: Iconos y descripciones	0,95	
O1: Esfuerzo	1	1
R1: Tiempo de respuesta	0	0
U1: Recursos utilizados	0	0
D1: Documentación fuentes	1	1
C1: Registro de cambios	1	1
M1: Estabilidad del sistema	1	1
G1: Facilidad de pruebas de aceptación	1	1
L1: Multiplataforma	1	1
H1: Adaptación al cambio	1	1

Fuente: autor

Para llevar a cabo la evaluación de la plataforma web de anuncios publicitarios, fue necesario la utilización de una metodología para el desarrollo del proceso y un encuesta de satisfacción por parte de un conjunto de usuarios, debido a que el análisis de las características del modelo de calidad de la ISO/IEC 9126 es un proceso complejo y el resultado de dicha evaluación debe ser totalmente confiable. De acuerdo a las deducciones obtenidas en la tabla 28, las características analizadas del modelo de calidad de la ISO/IEC 9126 los resultados tienen una evaluación alta (A). Para los tiempos de respuesta y recursos utilizados se le asignó una valoración de cero (0), teniendo en cuenta que dichas características se evalúan en un ambiente de producción.

5.7.4. Pruebas al sistema

Las pruebas realizadas a las funcionalidades del sistema permiten constatar que la información dada en la recopilación de la información y las historias de usuario hayan sido complemente cumplidas. Por eso se diseñaron algunas pruebas de aceptación para verificar el cumplimiento de los requisitos. La prueba de aceptación de la tabla 29 muestra el resultado del estudio realizado para la gestión de anuncios publicitarios.

Tabla 29: Prueba de aceptación 01

Prueba de aceptación 01			
Nombre	Anuncios		
Descripción	Evaluar la gestión de anuncios dentro del sistema		
Elemento	Actividades	Estado	Observaciones
Anuncios	Registrar anuncio	OK	
	Ver anuncios	OK	
	Actualizar anuncios	OK	
	Deshabilitar anuncios	OK	
Sucursal	Registrar sucursal	OK	
	Actualizar sucursal	OK	
	Ver sucursal	OK	
	Deshabilitar sucursal	OK	
Categoría	Registrar categoría	OK	
	Actualizar categoría	OK	
	Ver categorías	OK	
	Eliminar categorías	OK	

Fuente autor

La prueba de aceptación de la tabla 30 muestra el resultado del análisis realizado para la gestión de empresas en el sistema de anuncios publicitarios. Para la gestión de empresas es necesario indicar que también se realiza la gestión de contactos y la gestión de los lugares donde se encuentran ubicadas las empresas. Ver la tabla 30

Tabla 30: Prueba aceptación 02

Prueba de aceptación 02			
Nombre	Empresas		
Descripción	Evaluar la gestión de empresas dentro del sistema		
Elemento	Actividades	Estado	Observaciones

Empresa	Registrar empresa	OK	
	Ver empresas	OK	
	Actualizar empresas	OK	
	Deshabilitar empresas	OK	
Contacto	Registrar contacto	OK	
	Actualizar contacto	OK	
	Ver contactos	OK	
	Deshabilitar contactos	OK	
Lugar	Registrar lugar	OK	
	Actualizar lugar	OK	
	Ver lugares	OK	
	Eliminar lugares	OK	

Fuente: autor

Las pruebas de aceptación se aplican a un proyecto de software en una lista de chequeo para verificar que la totalidad de los requisitos de software dados por que cliente se hayan implementado en el producto creado. Para el proyecto de anuncios publicitarios (spot.co) se puede constatar que se cumple en su totalidad con la implementación de los requisitos del sistema.

6. Conclusiones

La metodología de desarrollo de software orientado por pruebas (TDD), es un marco de trabajo que se centra en el diseño de pruebas unitarias, teniendo como base las historias de usuario del producto. Se trabaja con metodologías agiles basadas en los procesos como SCRUM o Programación Extrema y permite entender claramente las funcionalidades del sistema para mejorar la calidad del producto, pero que al inicio del proceso, somete la productividad del equipo desarrollador, debido a la compleja tarea de crear, probar y refactorizar las pruebas unitarias, pero que se soluciona al adquirir experiencia en el proceso.

La implementación del estándar de la ISO/IEC 9126 a la plataforma de anuncios publicitarios (spot.co) y la encuesta por parte de un target poblacional de usuarios para apoyar el proceso muestra un resultado óptimo, lo que permite deducir que los productos desarrollados bajo la metodología de desarrollo de software orientado por pruebas (TDD) cuenta con un nivel alto de calidad. Sin embargo, cabe mencionar que dicho marco de trabajo se centra en el diseño del sistema y la lógica de negocio, mientras que las características del estándar de la ISO/IEC 9126 evalúan aspectos más amplios del sistema, pero que al igual tienen alguna relación importante con el diseño que aplica el modelo TDD.

La implementación de la metodología de desarrollo de software orientado por pruebas (TDD) a proyectos pequeños, como el de anuncios publicitarios de la presente investigación es eficiente y muestra un nivel de calidad optimo, sin embargo, es necesario implantar el modelo a proyectos de diferentes tamaños, para determinar si dicho marco de trabajo puede ser utilizado para este tipo de proyectos, teniendo en cuenta la importancia de la calidad del producto.

Las pruebas unitarias que se realizan como parte de la metodología desarrollo orientado por pruebas (TDD) permiten definir claramente la lógica de negocio del software a desarrollar, dando una pincelada inicial de lo que podrían ser las fuentes del proyecto final; pero al terminar la implementación del algoritmo TDD, dichas pruebas unitarias creadas, pasan a ser parte de las funciones del proyecto, y tienden a ser modificadas, debido a la captura de información de los formularios, validaciones y formatos de la información.

7. Trabajos futuros

El desarrollo del presente proyecto tuvo en cuenta la implementación de la metodología de desarrollo de software orientado por pruebas (TDD) a un proyecto de anuncios publicitarios (spot), teniendo en cuenta el procedimiento de dicho marco de trabajo y la evaluación del producto de acuerdo a las características del modelo de calidad de la ISO/IEC 9126, pero existen otros aspectos importantes a desarrollar más adelante como parte del desarrollo del presente proyecto. A continuación, se muestran algunas líneas de trabajo:

- Desarrollar e implementar una aplicación móvil de anuncios publicitarios, utilizando la metodología de desarrollo de software orientado por pruebas (TDD).

- Desarrollar e implementar proyectos de software de diferentes tamaños, utilizando la metodología de desarrollo de software orientado por pruebas (TDD) y evaluar los sistemas utilizando el modelo de calidad de la ISO/IEC 9126.

- Desarrollar modelo de calidad, que permita evaluar un producto de software desde las fases tempranas del proyecto hasta la terminación completa del producto.

- Desarrollar e implementar metodología de desarrollo de software orientado por pruebas (TDD) a la interface gráfica de usuario del proyecto y/o formularios de los sitios web.

- Metodología para la implementación del marco de trabajo, desarrollo de software orientado por pruebas (TDD) a las empresas de desarrollo de sistemas que manejen procesos diferentes a lo que representa TDD.

8. Bibliografía

[1] M. C. P. P. L. José H. Canós, “Métodologías Ágiles en el Desarrollo de Software,” Mar. 2012.

[2] L. E. SPARC (Organization), Eliécer; Valencia Ayala and Universidad Tecnológica de Pereira., *DEL MANIFIESTO ÁGIL SUS VALORES Y PRINCIPIOS*, vol. XIII, no. 34. Universidad Tecnológica de Pereira, 2007.

[3] M. Universidad Nacional de Colombia. Sede de Medellín. Escuela de Ingeniería de Sistemas., Ailin; Rojas C., *Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo*, vol. 5, no. 2. Escuela de Ingeniería de Sistemas de la Facultad de Minas, Universidad Nacional de Colombia Sede-Medellín, 2008.

[4] Agustin Yagüe y Juan Garbajosa, *Comparativa práctica de las pruebas en entornos tradicionales y ágiles*. Asociación de Técnicos de Informática, 2009.

[5] D. Fucci, H. Erdogmus, B. Turhan, M. Oivo, and N. Juristo, “A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?,” *IEEE Trans. Softw. Eng.*, vol. 43, no. 7, pp. 597–614, Jul. 2017.

[6] I. Pablo Andrés Vaca, I. Calixto Maldonado, I. Claudia Inchaurrondo, I. Juan Peretti, I. María Soledad Romero, and I. Matías Bueno, “Test-Driven Development -Una aproximación para entender su utilidad en el proceso de desarrollo de Software.”

[7] A. Federation of American Societies for Experimental Biology., N. A. Anguiano, M. Samdarshi, J. D. N. Dionisio, B. G. Fitzpatrick, and K. D. Dahlquist, *Federation proceedings.*, vol. 30, no. 1 Supplement. Federation of American Societies for Experimental Biology, 2016.

[8] D. Clerissi, M. Leotta, G. Reggio, and F. Ricca, “Test Driven Development of Web Applications: A Lightweight Approach,” in *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 2016, pp. 25–34.

[9] “Considering rigor and relevance when evaluating test driven development: A systematic review,” *Inf. Softw. Technol.*, vol. 56, no. 4, pp. 375–394, Apr. 2014.

[10] Y. Rafique and V. B. Misic, “The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 835–856, Jun. 2013.

[11] N. C. Borle, M. Feghhi, E. Stroulia, R. Greiner, and A. Hindle, “Analyzing test driven development based on GitHub evidence,” Jun. 2016.

[12] A. P. Ress, R. de Oliveira Moraes, and M. S. Salerno, “Test-Driven Development as an Innovation Value Chain,” *J. Technol. Manag. Innov.*, vol. 8, pp. 19–20, 2013.

[13] P. A. Vaca *et al.*, “Test-Driven Development -Beneficios y Desafíos para el Desarrollo de Software,” pp. 1850–2792.

[14] D. A. Godoy, H. Kotynski, E. A. Belloni, E. O. Sosa, M. Gamonal, and N. Regueira, “Simulando Proyectos de Desarrollo de Software que incorporan Test Driven Development.”

[15] D. V. Castro Cruz and D. Vladimir, “Sistema de capacitación virtual para la implementación de la metodología de desarrollo de software Test Driven Devolopment.,” Oct. 2016.

[16] M. Fernando, G. Pinzón, J. Sebastián, and G. Sanabria, “Standard ISO/IEC 9126-3 application in the entity-relationship conceptual data model,” vol. 22, no. 35, pp. 113–125, 2013.

[17] “Aplicación del modelo ISO 9126 para la evaluación de un sistema de aprendizaje virtual Facultad de tecnología de la información.”

[18] J. R. Molina Ríos, N. M. Loja Mora, M. P. Zea Ordóñez, and E. L. Loaiza Sojos, “Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python,” *Rev. Latinoam. Ing. Softw.*, vol. 4, no. 4, p. 201, Sep. 2016.

[19] J. Galdríz-Rosquel, C. E. S. Toca, and J. A. M. Alfonso, “SISTEMA DE GESTIÓN PARA LA ADMINISTRACIÓN DE LA POLICLÍNICA DELTA DE CIEGO DE ÁVILA,” *Universidad&Ciencia*, vol. 3, no. 2, pp. 1–10, Mar. 2017.

[20] A. V. López, A. Sánchez, and G. A. Montejano, “Definición de Métricas de Calidad para Productos de Software,” Jun. 2016.

[21] H. A. Guío Ávila, “Evaluación de las características de un sistema de información con base en la norma ISO/IEC 9126-1,” *SIGNOS - Investig. en Sist. gestión*, vol. 5, no. 2, p. 33, Aug. 2015.

[22] A. E. Chavarría, S. B. Oré, and C. Pastor, “Aseguramiento de la Calidad en el Proceso de Desarrollo de Software utilizando CMMI, TSP y PSP,” *RISTI - Rev. Ibérica Sist. e Tecnol. Informação*, vol. 2016, no. 20, pp. 62–77, Dec. 2016.

[23] C. B. Reynoso, “Introducción a la Arquitectura de Software,” *Univ. Buenos Aires*, vol. 33, 2004.

[24] S. G. Guia, S. G. Campus, and S. G. Talento, “Arquitectura de Software.”

[25] M. W. Maier, D. Emery, and R. Hilliard, “Software architecture: introducing IEEE Standard 1471,” *Computer (Long. Beach. Calif).*, vol. 34, no. 4, pp. 107–109, 2001.

[26] E. Bascón Pantoja, “El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing,” *Rev. Acta Nov.*, vol. 2, no. 4, 2011.

[27] A. T. Espinosa, J. G. C. Sagredo, M. M. Reyes, and M. de L. L. García, “Automatización de la codificación del patrón modelo vista controlador (MVC) en proyectos orientados a la Web,” *Cienc. ergo-sum*, vol. 19, no. 3, pp. 239–250, 2012.

[28] M. de L. Universidad Autónoma del Estado de México. Coordinación General de Investigación y Estudios Avanzados., Jesús Gamaliel; Trueba Espinosa, Adrian; Martínez Reyes, Magally; López García, *Automatización de la codifcación del patrón modelo vista controlador (MVC)en proyectos orientados a la Web*, vol. 19, no. 3. UAEM, Coordinación General de Investigación y Estudios Avanzados, 2012.

[29] Y. D. Facultad de Ingeniería Civil del Instituto Superior Politécnico “José Antonio Echeverría” (Cujae) and Y. F. Romero, *Patrón Modelo-Vista-Controlador*, vol. 11, no. 1. 2012.

[30] J. H. Canós, P. Letelier, and M. C. Penadés, “Metodologías Ágiles en el desarrollo de Software,” *VIII Jornadas Ing. Softw. y Bases Datos, JISBD*, 2003.

[31] R. A. G. Bustos, “Métodos de desarrollo de software: El desafío pendiente de la estandarización,” *Theoria*, vol. 12, pp. 23–42, 2003.

[32] R. S. Pressman, *Ingeniería del Software: Un enfoque práctico*. Mikel Angoar, 1997.

[33] J. H. GALLEGO, C. A. RODRIGUEZ, and D. Ivan, “Métodos de Aprendizaje de Máquina para la Detección de Arritmias Cardıacas.”

[34] M. G. Estayno, G. N. Dapozo, L. R. Cuenca Pletsch, and C. L. Greiner, “Modelos y Métricas para evaluar Calidad de Software,” in *XI Workshop de Investigadores en Ciencias de la Computación*, 2009.

[35] A. O. Duarte and M. Rojas, “Las metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo,” *Rev. Av. en Sist. e Informática*, vol. 5, no. 2, pp. 159–171, 2008.

[36] J. Emilio Labra Gayo, D. Fernández Lanvin, J. Calvo Salvador, and A. Cernuda del Río, “Una Experiencia de aprendizaje basado en proyectos utilizando herramientas colaborativas de desarrollo de software libre.”

[37] José Ignacio Herranz Roldán, “TDD como metodología de diseño de software - Paradigma,” Madrid, p. 1, Jan-2011.

[38] A. Mishra, *IOS code testing : test-driven development and behavior-driven development with Swift*. Apress, 2017.

[39] D. Fucci and B. Turhan, “On the role of tests in test-driven development: a differentiated and partial replication,” *Empir. Softw. Eng.*, vol. 19, no. 2, pp. 277–302, Apr. 2014.

[40] I. J. Joskowicz, “Reglas y Prácticas en eXtreme Programming,” 2008.

[41] M. R. De Giusti, G. Villarreal, A. Sobrado, and A. J. Lira, “Estadísticas distribuidas y gestión de calidad en el marco de la Iniciativa de Enlace de Bibliotecas,” in *V Simposio Internacional de Bibliotecas Digitales*, 2009.

[42] S. M. Abrahão, O. Pastor, L. Olsina, and J. Fons, “Un Método para Medir el Tamañno Funcional y Evaluar la Calidad de Sitios Web.,” in *JISBD*, 2001, pp. 477–490.

[43] D. García León and A. Beltrán Benavides, “Un enfoque actual sobre la calidad del

software," *ACIMED*, vol. 3, no. 3, pp. 40–42, 1995.

[44] S. C. Carrión, P. L. González, and R. T. Tandazo, "Técnicas avanzadas de programación aplicadas a DDS: un nuevo enfoque," *Maskana*, vol. 6, no. Supl., pp. 29–35, Dec. 2015.

[45] M. A. Mascheroni and E. Irrazábal, "Framework para la creación y ejecución de pruebas automatizadas sobre servicios REST."

[46] J. M. Ruiz, C. Pacífico, and M. M. Pérez, "Clasificación y evaluación de métricas de mantenibilidad aplicables a productos de software libre," Aug. 2017.

[47] M. G. Estayno, G. N. Dapozo, L. R. Cuenca Pletsch, and C. L. Greiner, "Modelos y métricas para evaluar calidad de software," 2009.

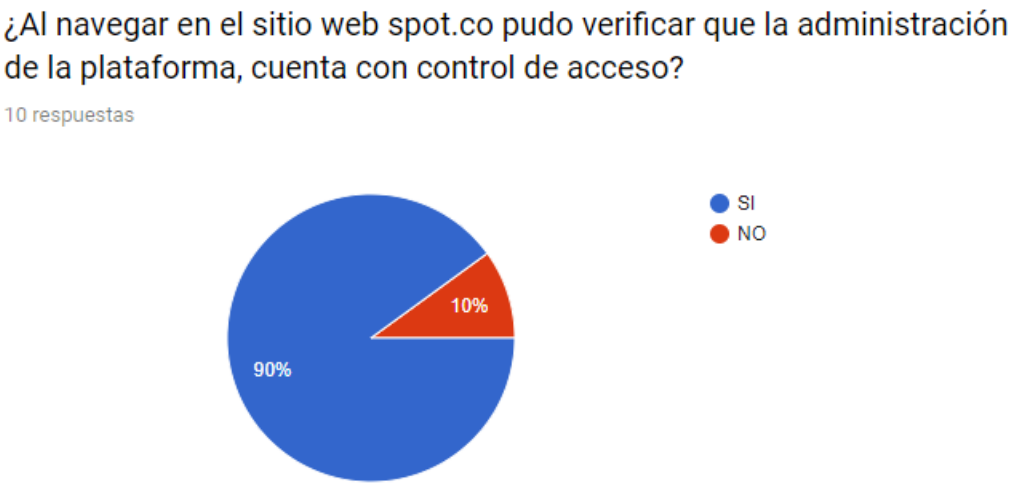
[48] S. J. Cochea Tomala and Espol, "Métricas de calidad de sistemas de información: aplicación en certificación de la calidad de un sistema en una empresa del sector hidrocarburifero," Nov. 2016.

[49] F. J. Díaz, C. M. Banchoff Tzancoff, A. S. Rodríguez, and V. Soria, "Herramientas open source para testing de aplicaciones web," 2009.

[50] E. S. Martínez, "Propuesta de procedimiento para realizar pruebas de caja blanca a las aplicaciones que se desarrollan en lenguaje Python," *3C TIC*, vol. 3, no. 2, pp. 89–114, Jun. 2014.

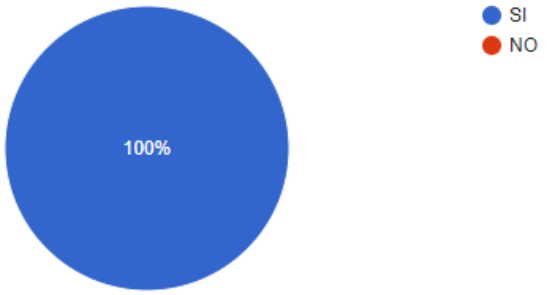
Anexos

Se anexan las gráficas de la encuesta realizada a un grupo de usuarios del sistema, para apoyar el proceso de evaluación de calidad de la plataforma de anuncios publicitarios (spot.co) de acuerdo a las características del estándar de la ISO/IEC 9126.



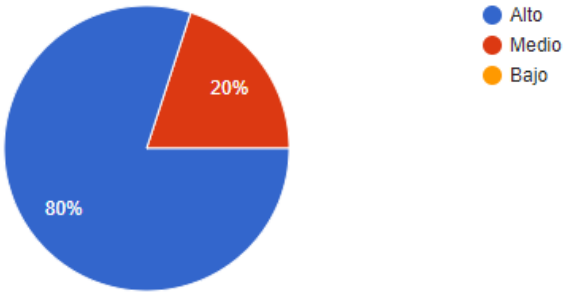
Spot.co por ser un directorio de anuncios publicitarios ¿le es fácil realizar las búsquedas de la información en la plataforma?

10 respuestas



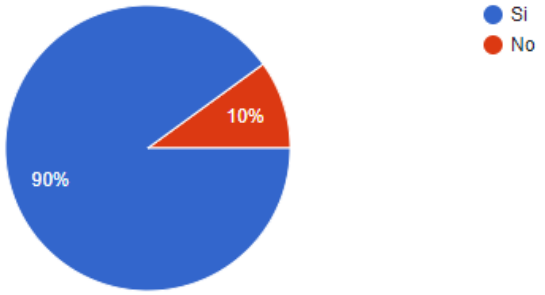
Al operar el sitio de anuncios publicitarios spot.co ¿Indique el nivel de facilidad de manejo que tiene dicho sitio web, de acuerdo a su consentimiento?

10 respuestas



¿Cree usted que las funciones que tiene la plataforma de anuncios spot.co actualmente, son exactas para lo que usted necesita?

10 respuestas



Artículo

A continuación se describe el artículo, como resultado de la investigación del presente proyecto. Se realiza un resumen de la implementación de la metodología de desarrollo de software orientado de pruebas y la evaluación con el modelo de calidad de la ISO/IEC 9126.

Aplicación y evaluación de la metodología desarrollo orientado por pruebas (TDD).

Oscar Iván Torres Corredor*

Máster Universitario en Ingeniería del Software y Sistemas Informáticos, Universidad Internacional de la Rioja, Departamento de Ingeniería, Logroño, España, oscartorrescorredor@gmail.com

Resumen

La metodología de desarrollo de software orientado por pruebas (TDD) está tomando fuerza en el delineamiento y desarrollo de sistemas; su facilidad de implementación y calidad del producto lo hace un modelo excelente de adoptar por las empresas. En este trabajo se propone el desarrollo de una plataforma de anuncios publicitarios, creada bajo la metodología TDD, evaluando la calidad del producto, teniendo en cuenta las características del modelo de calidad del estándar de la ISO/IEC 9126, para determinar el nivel de aptitud de la plataforma, y dar constancia de que dicha metodología puede ser adoptada en el desarrollo de software por las empresas. Para la investigación se manejaron los marcos de trabajo XP (programación extrema), TDD (desarrollo orientado por pruebas) y el estándar de la ISO/IEC 9126. La implementación del modelo de calidad al software de anuncios publicitarios (spot) muestra un

resultado prometedor en cuanto a las características evaluadas, sin embargo al finalizar la investigación se encontró que las pruebas de la metodología TDD solo funcionan en la capa de negocio, más no en la interfaz de usuario del software.

Palabras Clave: TDD, ISO/IEC9126, Calidad, Software, Modelo, Metodología.

I. Introducción

Las metodologías agiles son actualmente el tema de moda en lo que respecta a la ingeniería del software; desde que se creó el manifiesto ágil en el año 2001 [2] han surgido diferentes metodologías, que permiten facilitar el desarrollo de software, teniendo en cuenta los entornos cambiantes de las necesidades de los clientes y la calidad del producto. Sin embargo, son tantas los marcos de trabajo que se han creado, que las diferentes empresas solo han podido implementar algunos de ellos; teniendo dentro de los más utilizados, las metodologías de Programación Extrema, Scrum, Kanban y Agile Inception [3] [4], pero no se ha analizado profundamente los beneficios que puede brindar las demás metodologías agiles, para ser implementadas en el desarrollo de sistemas informáticos. La presente investigación se realizó con el fin de profundizar en el marco de trabajo de desarrollo orientado por pruebas (TDD) y la evaluación de calidad del producto bajo las características del estándar de la ISO/IEC 9126, para determinar el nivel de confianza del software desarrollado, bajo dicha metodología. Como conclusión, el marco de trabajo TDD, mejora la calidad del producto, pero compromete la productividad del equipo desarrollador, debido a la cantidad de pruebas que se deben realizar como parte del proceso algorítmico, que propone dicha metodología. Sin embargo, la implementación del modelo de calidad de la ISO/IEC 9126 al producto, muestra un nivel de calidad óptimo.

II. Contextualización

A continuación se describe las teorías en lo que respecta la investigación del presente proyecto, primero la metodología agile del desarrollo de software orientado por pruebas (TDD) y el modelo de calidad de software del estándar de la ISO/IEC 9126.

A. Metodología de desarrollo orientado por pruebas (TDD)

La metodología TDD (Desarrollo orientado por pruebas) es una técnica de diseño e implementación de software que hace parte del marco de trabajo XP (Programación extrema). [1] TDD define su proceso en escribir primero las pruebas del requisito del sistema, después implementar el código fuente que pase satisfactoriamente las pruebas del paso anterior, y

finalmente refactorizar el código escrito para eliminar la duplicidad y hacer mejoras. Una vez el desarrollador tiene claro la funcionalidad del software, lo expresa en código; al tener el ejemplo escrito, se codifica lo mínimo necesario para que se cumpla y la prueba pase de forma correcta. Finalmente, se modifica el diseño sin alterar su comportamiento, en búsqueda de líneas duplicadas; se revisa que el código cumpla con ciertos principios de diseño, para así determinar la calidad de las fuentes del proyecto. [37]

a. Ciclo de vida

El ciclo de vida que propone la metodología consiste en los siguientes pasos: el cliente escribe su historia de usuario, se escriben los criterios de aceptación junto con el cliente, desglosándolos para simplificarlos todo lo posible, se escoge el criterio de aceptación más simple y se traduce en una prueba unitaria, se comprueba que la prueba falla, se escribe el código que hace pasar la prueba, se ejecutan todas las pruebas automatizadas, se refactoriza y se limpia el código, se vuelven a pasar todas las pruebas automatizadas para comprobar que todo sigue funcionando, se vuelve con los criterios de aceptación que falten y se repite el ciclo una y otra vez hasta terminar la aplicación. [38]

B. Estándar de calidad ISO/IEC 9126

La ISO/IEC 9126 es un estándar de calidad de software, que evalúa las aplicaciones web desde diferentes puntos de vista, facilitando un conjunto de métricas que se pueden estimar cualitativamente o cuantitativamente para medir el nivel óptimo de las aplicaciones. El estándar de calidad reúne características que valoran los procesos desde la perspectiva interna y externa de las plataformas web, que a su vez permiten evaluar dichas aplicaciones desde el punto de vista de la funcionalidad, es decir el cumplimiento de requisitos del sistema; desde la fiabilidad, la flexibilidad del software para escalar; desde la usabilidad, el nivel de esfuerzo que realizan los usuarios finales para aprender a utilizar la herramienta; desde la eficiencia, es decir el rendimiento y tiempos de respuesta; desde la mantenibilidad, la capacidad de recuperarse y operar en el tiempo; y desde la portabilidad, es decir la capacidad de operar en diferentes plataformas. .

III. Desarrollo de la investigación

En el ciclo de vida del desarrollo de un proyecto de software el primer paso a concretar con el cliente, son las funcionalidades que debe tener el sistema. La metodología XP y SCRUM se centran en la escritura de pequeñas unidades conocidas como historias de usuario, para entender claramente la lógica de negocio. Ver figura 1

Tabla 1: Historia de usuario: HU01 – agregar anuncio

Historia de usuario – Sistema de información	
Número: HU01	**Nombre**: Agregar anuncio
Descripción: el sistema permitirá registrar nuevos anuncios en la plataforma, para que se puedan visualizar en el sitio web. Los datos que se deben proporcionar para un nuevo anuncio son: ▪ Fecha de publicación ▪ Fecha de terminación ▪ Tipo de Anuncio ▪ Descripción ▪ Sucursal ▪ Categoría	
Observaciones: para registrar un anuncio en el sistema, el usuario debe estar autenticado como *Administrador*.	

Fuente: autor

Al definir la historia de usuario, se implementa el algoritmo TDD al proyecto de software; primero se escribe una prueba unitaria que falle, seguidamente una prueba que pase correctamente y por último la refactorización de dicho código.

```
'''
Autor           Oscar Iván Torres Corredor
Fecha           10/12/2017
Descripción     Prueba unitaria: Guardar datos
'''
def testSaveAnuncios(self):
    categoria = Categoria.objects.create(nombre = "TAXIS")
    lugar = Lugar.objects.create(nombre = "Colombia", tipo = "P")
    empresa = Empresa.objects.create(nombre = "Empresa", slogan = "Es mejor", categoria = categoria)
    sucursal = Sucursal.objects.create(direccion = "Cra 1 N 10 - 50", empresa = empresa, lugar = lugar)
    anuncio = Anuncio.objects.create(descripcion = "Ejemplo de anuncio publicitario para el proyecto.", sucursal= sucur
    self.assertTrue(anuncio)
```

Figura 1: Prueba unitaria (autor)

De acuerdo al proceso de implementación del marco de trabajo TDD, la prueba que se implemente en las fuentes del proyecto debe fallar como primera medida, para luego realizar la refactorización y corregir los inconvenientes presentados. Al implementar y ejecutar la prueba unitaria del paso anterior, se puede observar el error en la prueba que permite registrar anuncios.

```
(venv) C:\Users\USER\Documents\advertising>python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..E..............
======================================================================
ERROR: testSaveAnuncios (main.tests.AnuncioTestCase)
----------------------------------------------------------------------
```

Figura 2: Fallo de prueba unitaria (autor)

Al fallar la prueba, como lo muestra la figura 2, se procede a modificar el código, para que la prueba sea correcta y se realiza la refactorización de dichas fuentes. Al finalizar las pruebas unitarias y entender ampliamente las funcionalidades de la plataforma de anuncios publicitarios, se procede a codificar el software. La metodología TDD precisa en sus procesos, que al contar con las pruebas unitarias totalmente correctas y organizadas, dichos códigos son traslados como funcionalidades de las fuentes finales del proyecto.

A. Características ISO/IEC 9126 para evaluar el software

A continuación se describen las características que se tuvieron en cuenta para realizar la implementación del estándar de la ISO/IEC 9126 al sistema desarrollado, teniendo en cuenta los procesos del marco de trabajo TDD. Para la implementación del modelo, se tuvo en cuenta la investigación [16] y así definir el procedimiento.

Tabla 2: Características del estándar de la ISO/IEC 9126

ISO/IEC 9126		CARACTERÍSTICAS
Funcionalidad	Idoneidad	I1: Nombre de etiquetas
		I2: Nombre de títulos
	Exactitud	E1: Exactitud de las funciones
	Interoperabilidad	P1: Interacción con otro sistema
	Seguridad	S1: Inicio de sesión
		S2: Control de acceso
		S3: Redirección a sitio autorizado
Fiabilidad	Tolerancia a fallos	T1: Manejo de errores
Usabilidad	Facilidad de comprensión	F1: Facilidad de acceso al buscador
		F3: Uso de colores
	Facilidad de aprendizaje	A1:Mensajes de ayuda
		A2: Multimedia
		A3: Imágenes
		A4: Iconos y descripciones
	Operatividad	O1: Esfuerzo
Eficiencia	Tiempo de uso	R1: Tiempo de respuesta
	Recursos utilizados	U1: Recursos utilizados
Mantenibilidad	Simplicidad de análisis	D1: Documentación fuentes
	Simplicidad de cambio	C1: Registro de cambios
	Estabilidad	M1: Estabilidad del sistema
	Facilidad de prueba	G1: Facilidad de pruebas de aceptación
Portabilidad	Facilidad de instalación	L1: Multiplataforma
	Facilidad de adaptación al cambio	H1: Adaptación al cambio

Fuente: autor

B. Resultado implementación norma ISO/IEC 9126

A continuación se muestra el resultado del análisis e implementación de la norma ISO/IEC 9126 al software desarrollado bajo el marco de trabajo TDD. La tabla 3 muestra los pesos que se le asignaron a cada característica del modelo de calidad. De acuerdo a la metodología consultada [16], la evaluación se realiza de manera cualitativa, sin embargo , se maneja una escala de 0 a 1, repartiendo las valoraciones de la siguiente manera:

Bajo (B): 0,00 a 0,33; Medio (M): 0,34 a 0,75; Alto (A): 0,76 a 1,00

Tabla 3: Resultados implementación estándar ISO/IEC 9126

CARACTERÍSTICAS	PUNTAJE	TOTAL CARACTERÍSTICA
I1: Nombre de etiquetas	0,95	0,935
I2: Nombre de títulos	0,92	
E1: Exactitud de las funciones	1	1
P1: Interacción con otro sistema	1	1
S1: Inicio de sesión	1	1
S2: Control de acceso	1	
S3: Redirección a sitio autorizado	1	
T1: Manejo de errores	0,94	0,94
F1: Facilidad de acceso al buscador	1	0,975
F3: Uso de colores	0,95	
A1:Mensajes de ayuda	0,96	0,94
A2: Multimedia	0,92	
A3: Imágenes	0,93	
A4: Iconos y descripciones	0,95	
O1: Esfuerzo	1	1
R1: Tiempo de respuesta	0	0
U1: Recursos utilizados	0	0
D1: Documentación fuentes	1	1
C1: Registro de cambios	1	1
M1: Estabilidad del sistema	1	1
G1: Facilidad de pruebas de aceptación	1	1
L1: Multiplataforma	1	1
H1: Adaptación al cambio	1	1

Fuente: autor

Para llevar a cabo la evaluación de la plataforma web de anuncios publicitarios, fue necesario la utilización de una metodología para el desarrollo del proceso, debido a que el análisis de las características del modelo de calidad de la ISO/IEC 9126 es complejo y el resultado de dicha evaluación debe ser totalmente confiable. De acuerdo a los resultados obtenidos en la tabla 2, las características analizadas del modelo de calidad de la ISO/IEC 9126 los resultados tienen una evaluación alta (A). Para los tiempos de respuesta y recursos utilizados se le asignó una valoración de cero (0), teniendo en cuenta que dichas características se evalúan en un ambiente de producción.

IV. Conclusiones

La metodología de desarrollo de software orientado por pruebas (TDD), es un marco de trabajo que se centra en el diseño de pruebas unitarias, teniendo como base las historias de usuario del producto. Se trabaja con metodologías agiles basadas en los procesos como SCRUM o Programación Extrema y permite entender claramente las funcionalidades del sistema para mejorar la calidad del producto, pero que al inicio del proceso, somete la productividad del equipo desarrollador, debido a la compleja tarea de crear, probar y refactorizar las pruebas unitarias, pero que se soluciona al adquirir experiencia en el proceso.

La implementación del estándar de la ISO/IEC 9126 a la plataforma de anuncios publicitarios (spot.co) y la encuesta por parte de un target poblacional de usuarios para apoyar el proceso muestra un resultado óptimo, lo que permite deducir que los productos desarrollados bajo la metodología de desarrollo de software orientado por pruebas (TDD) cuenta con un nivel alto de calidad. Sin embargo, cabe mencionar que dicho marco de trabajo se centra en el diseño del sistema y la lógica de negocio, mientras que las características del estándar de la ISO/IEC 9126 evalúan aspectos más amplios del sistema, pero que al igual tienen alguna relación importante con el diseño que aplica el modelo TDD.