UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

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

“Módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas.”

Autor:

Pedro Antonio García González

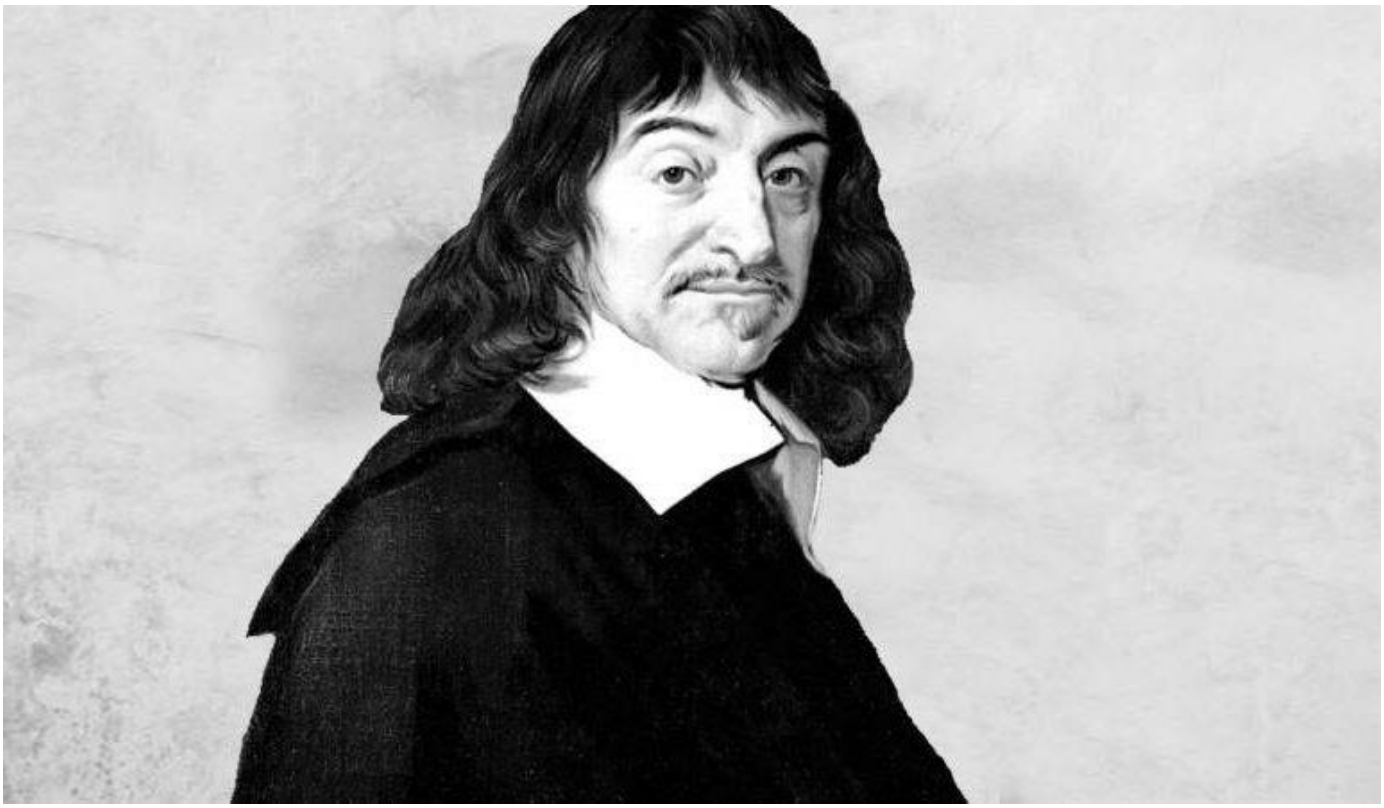
Tutores:

Msc. Joaquín Quintas Santiago

Ing. Ilsen León Herrera

La Habana, noviembre de 2018

¨Año 60 de la Revolución¨



Daría todo lo que sé, por la mitad de lo que ignoro (René Descartes)

Declaración de autoría

Me declaro como único autor de la presente tesis y reconozco a la Empresa de Tecnologías de la Información para la Defensa(XETID) como entidad con los derechos patrimoniales exclusivos sobre la misma.

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

Pedro Antonio García González

_____

Firma del Autor

_____ _____

Ing. Ilsen León Herrera Msc. Joaquín Quintas Santiago

Dedicatoria

Con el siguiente trabajo se otorga el Título Ingeniería en Ciencias Informáticas, a Aida María González Amador y Pedro José García Fuentes, quédenselo, yo aprendí todo lo que necesitaba cuando empecé a pronunciar palabras como: mamá y papá.

Agradecimientos

A lo(s) José Antonio García, tío y abuelo, que, si bien no están ya entre nosotros, tienen gran culpa de los logros alcanzados, culpables de todo lo que hoy soy y tengo, viejos gracias por el domino y los domingos juntos en los que el mundo era perfecto.

A Aida María y Pedro José a los que comúnmente llamo mamá y papá. Gracias porque sé que tengo más de lo que merezco y es gracias a ustedes. Comprendes el verdadero valor de lo que has logrado cuando haces sonreír y vez orgullo en los ojos de quienes te trajeron aquí. Yo no soy ateo, yo creo en ustedes.

A toda la familia que lejos o cerca de mí siempre estuvieron al pendiente y me apoyaron a lo largo del camino para poder lograrlo.

A todas las amistades que nacieron en esta escuela y vivirán fuera de ella, a la gente del 4106, muchos años juntos caballero, gracias Luisma y Leyan, ellos saben porque en esta lista están.

A todos los profesores que contribuyeron en mi formación a lo largo de la carrera, gracias por el apoyo y el tiempo dedicado.

A la KANOA, al grupo de amigos que un día se unió por un nombre y objetivo en común, y que un día como hoy entiendes que no es la meta es el camino lo que te hace diferente, gracias mi gente por el parque y la escalera y por entender que familia también nace en la escuela.

Resumen

El sector de las reparaciones domésticas ha alcanzado en los últimos tiempos un nivel de demanda elevado en Cuba, unido al aumento de esta demanda y gracias a los cambios en el modelo económico realizados en el país, también se ha elevado el volumen de personas principalmente en el sector no estatal que se dedican a esta actividad tan necesaria. De la mano con este aumento viene asociada la falta de una entidad o sistema que permita escoger al trabajador que más se adecue a las necesidades de cada cliente.

El presente trabajo permitió el desarrollo del Módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas empleando arquitectura de micro-servicios para la plataforma D'Prisa. El mismo garantiza la interrelación entre los clientes y proveedores del sector de las reparaciones domésticas, simplificando el proceso de contratación de estos servicios. Para ello, se utilizó como framework de desarrollo angular, aplicando la metodología de desarrollo de software eXtreme Programming (XP) y gestor de base de datos postgreSql. Se realizaron pruebas unitarias, de aceptación y de integración, arrojando resultados satisfactorios.

Palabras claves:

Reparaciones domésticas, módulo, clientes, micro-servicios.

Abstract

The domestic repair sector has recently reached a high level of demand in Cuba, together with the increase in this demand and thanks to the changes in the economic model carried out in the same country, there has also been an increase in the volume of people mainly in the non-state sector who dedicate themselves to this much-needed activity. Hand in hand with this increase is the lack of an entity or system that allows the worker to be chosen that best suits the needs of each client.

The present work allowed the development of the module of clients of domestic repair services using the architecture of micro-services for the D'Prisa platform. It guarantees the interrelation between clients and suppliers of the domestic repair sector, simplifying the process of contracting these services. For this purpose, it was used as an angular development framework, applying the eXtreme Programming (XP) software development methodology and postgreSql database manager. Unit, acceptance and integration tests were carried out, with satisfactory results.

Keywords:

Domestic repairs, module, clients, micro-services.

Índice general

Índice de Tablas

Índice de Figuras

Introducción

Hoy en día la continua actualización o cambio del modelo económico, son elementos necesarios para la supervivencia económica de cualquier país. En muchos países estas transformaciones han permitido un crecimiento del ingreso per cápita, lo que acarrea una mejora en los niveles de vida de los habitantes del país, evidenciándose así la importancia de dicho proceso. Cuba en su plan de actualización del modelo económico reconoce la importancia de otras formas de gestión, paralelas a la gestión estatal, para garantizar la productividad y la eficiencia en aras de mantener los principios y conquistas de la nación.

Actualmente el mercado cubano se ha visto invadido por este sector no estatal y debido a la gran variedad de personas que practican el trabajo por cuenta propia se tiene actualmente una serie de servicios que varían en disímiles factores, destacándose entre estos el factor calidad y precio, lo que ha dado a las personas que consumen los servicios por cuenta propia la tarea de buscar al cuentapropista que más se ajuste a sus posibilidades y que a la vez satisfaga sus necesidades.

Prueba irrefutable de esta situación se encuentra en el sector de las reparaciones domésticas siendo este una porción que toca personalmente a todos, ya que en la mayoría de hogares del mundo siempre en algún momento determinado se necesita realizar alguna reparación o modificación de los mismos, y cuando eso sucede lo primordial en nuestro país es encontrar al cuentapropista correcto, lo que nos deja con la gran tarea de encontrarlo, esta tarea se ve obstruida porque la información relativa a los proveedores es escaza y no existe un sistema especializado en almacenar y difundir dicha información, trayendo como consecuencias:

Del lado del cliente:

- Dificultad para localizar los proveedores.
- Dificultad para seleccionar un proveedor determinado.
- Perdida adicional de tiempo y dinero.
- Dificultad para divulgar la calificación del servicio obtenido de un proveedor determinado.

Del lado del proveedor:

- Dificultad para divulgar su trabajo.
- Falta de ofertas de trabajo disponibles.

Dada esta situación y el auge que ha tenido el uso de las TIC en Cuba y en el mundo, era de esperarse que alguna entidad dedicada a la informatización de la sociedad cubana tomara de la mano esta problema, este es el caso de la empresa de Tecnologías de la Información para la Defensa (XETID) , dedicada al sector del

software, la automática y las comunicaciones y que tiene entre sus principales objetivos la producción de soluciones informáticas que cumplan con los estándares nacionales e internacionales y que posean un alto nivel de calidad. Esta empresa ha estudiado la oportunidad de incluir en una de sus plataformas de desarrollo un sistema para darle solución a este problema, la plataforma escogida es D´Prisa, que tiene la peculiaridad de poseer una arquitectura basada en micro-servicios, los cuales son comúnmente utilizadas en los últimos tiempos debido a su amplia gama de ventajas entre las que se destaca su flexibilidad a la hora de crearlos y desplegarlos.

Teniendo en cuenta la situación antes expuesta, surge el siguiente **problema a resolver:** ¿Cómo contribuir al proceso de reseña, clasificación y contratación de servicios de reparaciones domésticas?

Definiendo como **objeto de estudio:** Sistemas para la reseña, clasificación y contratación de servicios.

Para dar solución al problema planteado se define como **objetivo general**: Desarrollar el Módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas empleando arquitectura de micro-servicios para cuentapropistas sobre la plataforma D´Prisa.

Teniendo como **campo de acción:** Sistemas para la reseña, clasificación y contratación de servicios de reparaciones domésticas para cuentapropistas.

En función de cumplir con el objetivo general se trazaron los **objetivos específicos**

siguientes:

- Establecer el marco teórico-conceptual para fundamentar la investigación a partir del estudio de sistemas reseña, clasificación de servicios y contratación de reparaciones domésticas.
- Realizar el análisis y diseño del módulo de clientes para la reseña, clasificación de servicios y contratación de reparaciones domésticas para facilitar la implementación del sistema.
- Implementar el módulo de clientes para la reseña, clasificación de servicios y contratación de reparaciones domésticas para facilitar la implementación del sistema.
- Realizar el proceso de pruebas a la solución propuesta.

Para cumplir con los objetivos mencionados anteriormente se proponen un grupo de **tareas de la investigación** que servirán de guía, las cuales son:

- Analizar los conceptos fundamentales implicados en el proceso de reseña, clasificación de servicios y contratación de reparaciones domésticas.
- Realizar el estudio del estado del arte sobre los sistemas de reseña, clasificación de servicios y contratación de reparaciones domésticas.

- Seleccionar las tecnologías, herramientas, metodología y lenguaje necesarios para el desarrollo del módulo.
- Realizar el proceso de identificación, especificación y validación de los requisitos funcionales y no funcionales del sistema.
- Diseño e implementación de la propuesta de solución para dar solución al problema existente.
- Realizar el proceso de pruebas a la solución propuesta.

Para el desarrollo satisfactorio del presente trabajo de diploma se emplearon métodos científicos tanto teóricos como empíricos, los cuales se mencionan a continuación:

Métodos Teóricos

Método Histórico-Lógico

Es empleado en el estudio del arte del tema a investigar para conocer acerca de la existencia y características de sistemas de reseña, calificación y contratación de servicios de reparaciones domésticas que hayan sido creados anteriormente. Así como de la información asociada a las metodologías, herramientas y validaciones a usar en la solución.

Método Analítico-Sintético

Se emplea en el momento de buscar la esencia del tema en cuestión, también permite realizar análisis de teorías, de documentos, entre otros; realizando una síntesis de la misma posibilitando así encontrar los elementos más importantes que se relacionan con el proceso de diseño de sistemas informáticos de esta índole.

Métodos Empíricos

Observación

Con este método es posible distinguir directamente los hechos de la realidad objetiva. Permite conocer el proceso delimitado como objeto de estudio, lo cual contribuyó a tener un registro visual más detallado de lo que se quiere y hace falta hacer; y cómo hay que hacerlo. Mediante este método se analizaron varios sistemas de búsqueda de productos y se obtuvo una guía para el desarrollo del nuevo sistema.

Investigación documental

Se usó a la hora de llevar a cabo tareas que son claves para el desarrollo del sistema como la recolección, selección, análisis y presentación de información coherente a partir del uso de documentos. La realización de una recopilación adecuada de datos e información que permiten redescubrir hechos, sugerir problemas,

orientar hacia otras fuentes de investigación, orientar formas para elaborar instrumentos de investigación y elaborar hipótesis Considerarse como parte fundamental de un proceso de investigación científica, mucho más amplio y acabado.

Para facilitar la comprensión del documento se estructura de la siguiente forma:

En el **Capítulo 1** - Fundamentación teórica: se hace un análisis del estado del arte del objeto de estudio, se investiga acerca de los sistemas informáticos vinculados al campo de acción, se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo del sistema de gestión.

En el **Capítulo 2** - Características, análisis y diseño del sistema: En este capítulo se abordan las dos actividades estructurales iniciales de la metodología XP, las cuales son planificación y diseño, obteniendo de ellas artefactos como historias de usuario, la estimación de esfuerzos, el plan de iteraciones, las tarjetas CRC, entre otros los cuales describen una por una las funcionalidades del módulo a implementar.

En el **Capítulo 3** - Implementación y pruebas: En esta etapa se definen las tareas de ingenierías correspondientes a cada HU. También se aplican las pruebas unitarias, pruebas de integración y pruebas de aceptación que permiten comprobar el correcto desarrollo de las funcionalidades implementadas.

Capítulo 1: Fundamentación teórica

En el presente capítulo se abordan los principales conceptos relacionados con la problemática a resolver, se realiza un estudio de los diferentes sistemas existentes en el ámbito nacional e internacional relacionados con el tema a investigar, para ello se tienen en cuenta características que permiten comprender su funcionamiento y cuánto aportan a la investigación en curso. Además, se analizan algunas de las diferentes metodologías para el desarrollo de software, así como las tecnologías, herramientas y lenguajes que posibilitarán el desarrollo de la propuesta de solución.

1.1 Definición de los principales conceptos asociados al dominio del problema:

Micro-Servicios:

Una arquitectura de micro-servicios es un enfoque para desarrollar una aplicación software como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí. Normalmente hay un número mínimo de servicios que gestionan cosas comunes para los demás (como el acceso a base de datos), pero cada micro-servicio es pequeño y corresponde a un área de negocio de la aplicación. Además, cada uno es independiente y su código debe poder ser desplegado sin afectar a los demás. Incluso cada uno de ellos puede escribirse en un lenguaje de programación diferente, ya que solo exponen la API (una interfaz común, a la que le da igual el lenguaje de programación en la que el micro-servicio esté programado por debajo) al resto de micro-servicios. (Daniel López, 2017).

API:

Una API (siglas de 'Application Programming Interface') es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software (APIs, 2018).

MicroFrameworks

Microframework es el término que ha sido usado para referirse a Frameworks minimalistas de aplicaciones web, en contraste con Frameworks Full-Stack. Un microframework carece de la mayoría de funcionalidades que es común esperar en un Full-Stack, (como lo es Django) tales como manejo de cuentas, autenticación, autorizaciones, roles, permisos, abstracción de Bases de datos a través de un mapeo de objetos relacionales, motores de plantillas web, entre otras. Puntualmente lo que se busca al usar este microframework, es facilitar la recepción de peticiones HTTP, entregando la petición al controlador apropiado, despachar, y devolver una respuesta HTTP (Henao, 2018).

1.2 Plataforma D'Prisa:

Plataforma que tiene como objetivo principal el desarrollo rápido de aplicaciones a través de un espacio colaborativo donde los desarrolladores podrán compartir código listo para su empleo en la forma de módulos de software autónomos denominados "MicroServicios". Gracias a esta tecnología que se va imponiendo gradualmente en el mundo, se podrán crear aplicaciones distribuidas y escalables (Nativas de "La Nube").

Permitirá también aplicar la filosofía de "MicroFramework" donde, no se requerirá emplear un marco de trabajo completo para emplear un par de funciones sencillas. Se podrá seleccionar única y exclusivamente los elementos necesarios para desarrollar la aplicación. (Santiago, 2018).

Se propone, además, el desarrollo de "MicroFrontends" a partir de un conjunto de lineamientos que permitirán un desarrollo mucho más rápido de aplicaciones Web y aplicaciones Android, entre otros. Inicialmente constituirá una propuesta que será validada y enriquecida con la práctica y el uso diario. Otra de las funciones que pretende realizar D´Prisa es ofrecer una arquitectura básica de desarrollo de aplicaciones a partir de los conceptos mencionados que, si bien no es la solución para todo el universo de problemas que la informática tiene el deber de resolver, se puede acercar bastante a un 70% u 80% de los casos. (Santiago, 2018).

1.3 Análisis de Soluciones Existentes

Actualmente existen soluciones encargadas de la reseña, clasificación de servicios y contratación de reparaciones domésticas. Por tal razón se hace necesario realizar un estudio de las mismas, con el objetivo de definir si cubren las necesidades planteadas en la problemática descrita. A continuación, se realizará una breve descripción de cada uno con el objetivo de observar sus particularidades para ver si satisfacen las necesidades existentes.

1.3.1 Internacional

Reparaciones del Hogar y de Viviendas

Esta web cuenta un ágil sistema para solicitar presupuesto online gratuito, cuenta con una red de empresas de reparaciones y reformas que darán respuesta a las necesidades de cada cliente en averías o instalaciones que se presenten en su hogar, casa, piso, local o vivienda. Hogar soluciones cuenta con empresas de reparaciones y reformas integrales de alta cualificación. Su zona de operación es en España.

Fig. 1 Reparaciones del Hogar y de Viviendas

hogar-soluciones.es es un sistema informático destinado a prestar servicios de contratación de reparaciones domésticas y se accede a través de la url: **https://www.hogar-soluciones.es** , los principales servicios que brinda son:

- Reparaciones de Albañilería
- Servicios de pintura
- Reparaciones de Carpintería de Madera
- Servicios de Cerrajería
- Reparaciones de Cristalería
- Reparación de electrodomésticos

Hogar Reparación:

Es una web cuya base de datos dispone de pintores, cerrajeros, técnicos de calderas, de mantenimiento de piscinas, jardinería, e instaladores. Su web es intuitiva y sencilla además está bien definida y dotada de opciones donde incluso, tienes la posibilidad de subir una fotografía de la avería. Su zona de operación es en España principalmente en Madrid, Barcelona, Valencia, Bilbao, Zaragoza, Sevilla y Málaga, su arquitectura no está basada en los micro-servicios y su código es privado.

Fig. 2 Hogar Reparación

hogarreparacion.com es un sistema informático destinado a prestar servicios de contratación de reparaciones domésticas y se accede a través de la url: **https://www.hogarreparacion.com** , los principales servicios que brinda son:

- Servicios de Pintura
- Reparaciones de Carpintería
- Servicios de Fontanería
- Reparaciones de calderas y calefacción
- Reparación de electrodomésticos

Tele-técnicos

Empresa especializada en las reparaciones del hogar y el mantenimiento de comunidades y empresas. Ofrece soluciones las 24 horas del día, los 365 días del año. Cuenta con más de 15 años de experiencia en el negocio de las reparaciones y mantenimientos a domicilio.

Fig. 3 Tele-técnicos

teletecnicos.com es un sistema informático destinado a prestar servicios de contratación de reparaciones domésticas y se accede a través de la url: **https://www.teletecnicos.com** , los principales servicios que brinda son:

- Cerrajeros
- Electrodomésticos
- Fontaneros
- Electricistas
- Limpiezas
- Tejados

Sr. Handyman

El Sr. Handyman International, LLC es un negocio de franquicias, con sede en Ann Arbor, Michigan en los Estados Unidos, que ofrece servicios de mantenimiento y remodelación para propietarios y locales comerciales. Es una subsidiaria de The Dwyer Group. Lleva años en el mercado (desde el 2000) por lo que es una de las más robustas y confiables web que puedas encontrar si estás buscando hacer algún trabajo en tu hogar.

Esta web opera en países de habla inglés, principalmente en los Estados Unidos de América, aunque posee sucursales en otros países como Panamá, es de código privado y no posee arquitectura de micro-servicios.

Fig. 4 HandyMan Solutions

handymanpanama.com es un sistema informático destinado a prestar servicios de contratación de reparaciones domésticas y se accede a través de la url: **https://www.handymanpanama.com** , los principales servicios que brinda son:

- Plomería
- Carpintería
- Albañilería
- Telefonía
- Electrodomésticos
- Pintura y Acabados

HomeServe

A pesar de su nombre, no solo se dedica a realizar reparaciones, sino que a través de ella pueden conseguirse servicios a medida y resuelve incidencias concretas. Asimismo, incluye una sección propia de seguros de reparaciones que la propia entidad gestiona, y es muy sencilla de utilizar.

Esta web opera meramente en países de habla hispana de Europa principalmente en España, es de código privado y su arquitectura no se basa en los micro-servicios.

Fig. 5 Home Serve

homeserve.es es un sistema informático destinado a prestar servicios de contratación de reparaciones domésticas y se accede a través de la url: **https://www.homeserve.es** , los principales servicios que brinda son:

- Albañiles
- Fontaneros
- Electricistas
- Cerrajeros
- Técnicos del aire acondicionado
- Calefacción
- Pintores

1.3.2 Nacional

En el estudio de las soluciones existentes no se encontró en el ámbito nacional ningún sistema especializado en la reseña, calificación de servicios y contratación de reparaciones domésticas.

Resumen de soluciones existentes

Con el objetivo de una mejor comprensión del estudio realizado se resumen a continuación los sitios y sus características principales marcando con una ✔ las que posee y con una ✘ las que no posee.

Tabla 1: Resumen de soluciones existentes

Sistemas/Características		Hogar Reparación	Sr. Handyman	UrgeClick	HomeServe
Funcionalidades	Permite clasificar el servicio recibido	✗	✓	✓	✗
	Atención Rápida(en menos de 3 horas)	✓	✗	✓	✗
	Listar los servicios por varios criterios	✓	✓	✓	✓
	Servicio de chat	✗	✓	✗	✗
	Permite adjuntar una foto de la avería	✓	✓	✓	✓
	Brinda gran variedad de servicios	✓	✓	✓	✓
Aspectos de interés	Software privado	✓	✓	✓	✓
	Sitio intuitivo y sencillo	✓	✓	✓	✓
	Opera en Cuba	✗	✗	✗	✗

Los sistemas antes mencionados, de manera general, recogen los rasgos más significativos e importantes de los procesos de reseña, calificación de servicios y contratación de reparaciones domésticas, pero independientemente cada una presenta inconvenientes de uso. Estos sistemas son privativos y no tienen alcance en nuestro país, además no cuentan con la arquitectura de micro-servicios, necesaria para agregarlos a la plataforma D´Prisa. Estas condiciones anulan el interés de la empresa XETID en ellas, por lo que se decidió desarrollar una aplicación web que esté acorde a sus necesidades.

1.4 Metodología, tecnologías y herramientas a usar en el desarrollo de la solución:

Existen numerosos lenguajes de programación, herramientas y tecnologías que se pueden utilizar para la correcta elaboración del Módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas. Para el exitoso desarrollo de la solución al problema planteado es necesario realizar una profunda investigación de las metodologías, las tecnologías y herramientas que garanticen un producto que

responda a las necesidades requeridas y con la calidad que se merece el cliente, por lo que se propuso el estudio que se detalla a continuación.

1.4.1 Metodología de desarrollo de software

Una metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Las metodologías de desarrollo de software se pueden separar en dos grandes corrientes las metodologías tradicionales y las metodologías ágiles. Las tradicionales caracterizadas por su rigoroso control de los procesos y el seguimiento a las actividades involucradas en ellas, mientras que las metodologías ágiles, se caracterizan por hacer énfasis en la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde clientes y desarrolladores trabajan constantemente juntos, estableciéndose así una estrecha comunicación (Villegas, 2015).

No existen metodologías buenas o malas. Todas son bienintencionadas e intentan responder a problemas típicos de la gestión de proyectos. Sin embargo, se puede señalar que algunas son más eficaces, ágiles y acordes con determinados contextos, ahí radica su principal diferencia. No es una cuestión del fabricante o del impacto que tenga en el mercado, lo principal es su capacidad de respuesta (Villegas, 2015).

Para la correcta selección de la metodología a usar es necesario tener en cuenta indicadores como las capacidades y habilidades de desarrollo del equipo de trabajo, el compromiso del cliente, el tiempo de desarrollo, el trabajo en equipo, la disposición del equipo ante los cambios durante el proyecto y los recursos materiales disponibles. Es por ello que se decidió el uso de una metodología ágil para el desarrollo de la solución al problema planteado anteriormente porque se considera que responde a las necesidades y condiciones reales del equipo de desarrollo, teniendo en cuenta la importancia de la estrecha y constante interacción cliente-desarrollador.

Dentro del panorama ágil existen diferentes opciones. Programación Extrema y Scrum son las alternativas más conocidas, pero, cada una de ellas, cuenta con una forma diferente de entender la flexibilidad. Así, cada una de estas metodologías ágiles de gestión de proyectos se caracteriza por:

Programación Extrema:

La metodología XP o Programación Extrema es una metodología ágil y flexible utilizada para la gestión de proyectos. Se centra en potenciar las relaciones interpersonales del equipo de desarrollo como clave del éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo. Esta metodología pone el énfasis en la retroalimentación continua entre cliente y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes (Calvo, 2018).

Características

- Se considera al equipo de proyecto como el principal factor de éxito del proyecto
- Interacción constante entre el cliente y el equipo de desarrollo.
- Planificación flexible y abierta.
- Rápida respuesta a cambios.

Scrum:

La metodología SCRUM, es una metodología ágil y flexible utilizada para la gestión de proyectos. Es el orden dentro del caos, aceptando la naturaleza cambiante de un proyecto, trata de proponer directrices que simplifiquen su gestión. Ésta es, de todas las metodologías ágiles de gestión de proyectos, la que con mayor eficacia facilita el hallazgo de soluciones específicas para los problemas que van surgiendo durante el desarrollo del proyecto. Pese a sus ventajas, en especial en lo referente a la coordinación de personas, entre sus inconvenientes se encuentra el fomentar un entorno donde los niveles de estrés son difíciles de controlar, en parte debido a la necesidad de trabajar bajo presión para poder realizar las entregas parciales a tiempo.

Scrum descompone la organización en pequeños equipos auto-organizados. Cada equipo desarrolla los proyectos en base a entregas parciales “sprints”, con el objetivo de alinear expectativas con el cliente y aumentar el valor que se ofrece a los mismos. (Calvo, 2018)

Selección de la metodología

A partir de un análisis exhaustivo de las ventajas y limitaciones de metodologías de desarrollo de software como XP (extreme programming) y Scrum, en el que se tomaron en cuenta una serie de factores, como el tamaño del equipo, el nivel de adaptación a los cambios, la relación con el cliente y el nivel de documentación que genera, se tomó la decisión de seleccionar la metodología de desarrollo de software XP, teniendo como premisa las exigencias y características del nuevo sistema.

1.4.2 Lenguaje de modelado

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no el proceso que se siguió para obtenerlo. (Cornejo, 2008)

Lenguaje Unificado de Modelado

El lenguaje de modelado seleccionado para guiar el diseño de la aplicación fue el Lenguaje Unificado de Modelado (UML por sus siglas en inglés). UML es la sucesión de una serie de métodos de análisis y diseño orientado a objetos. Los métodos comprenden tanto el lenguaje de modelado como el proceso que se sigue para obtenerlo. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. (Cornejo, 2008)

Lo fundamental de una herramienta UML es la capacidad de diagramación, y los diferentes tipos de diagramas que soporta la herramienta. Sus esquemas de apoyo de diseño, documentación, construcción e implantación de sistema. Así mismo, su flexibilidad para admitir cambios no previstos durante el diseño o el rediseño. En resumen, la herramienta ideal, es aquella que admite diseño desde inicio a fin, diseño inverso (o rediseño) y diseño vise-versa, con esquemas amplios para documentar detalladamente los procesos. (Iyair Armando Díaz Sánchez, 2011).

Los principales beneficios de UML son:

- Modelar sistemas utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.4.3 Herramientas para el modelado

Herramienta CASE

Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas. Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales (Beltrán, 2012).

Visual Paradigm 8.0

Potente herramienta multiplataforma de diseño que proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad más rápido y mejor. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes, lo que supera a todo su proceso de desarrollo de Despliegue de Código de Modelo en esta solución integral (Visual Paradigm, 2009).

Entre sus características fundamentales se tienen:

- Multiplataforma: soportada en plataforma Java para Sistemas Operativos Windows, Linux, Mac OS.
- Interoperabilidad: intercambia diagramas y modelos con otras herramientas. Soporta la importación y exportación a formatos XMI y XML y archivos Excel. Permite importar proyectos de Rational Rose y la integración con Microsoft Office Vision.
- Integración con entornos de desarrollo: apoyo al ciclo de vida completo de desarrollo de software en IDEs como: Eclipse, Microsoft VisualStudio, NetBeans, Sun ONE, Oracle JDeveloper, Jbuilder y otros.

1.4.4 Tecnologías del lado del cliente

Las tecnologías del lado del cliente son aquellas que se ejecutan en el navegador del usuario, cargando páginas dinámicas que se procesan en el cliente.Dentro de las tecnologías del lado del cliente se destacan los lenguajes a utilizar los cuales son HTML 5, CSS3 y JavaScript.

HTML 5

Hyper Text Markup Language o Lenguaje de marcado de hipertexto en español (HTML5), es un lenguaje de etiquetas, utilizado para la estructuración y la presentación de contenido en los sitios web. Sus objetivos principales son mejorar el lenguaje dando soporte para los últimos objetos multimedia mientras se mantiene fácilmente legible por los humanos y a su vez ser comprendido constantemente por ordenadores y dispositivos como navegadores web y programas de análisis. A raíz de sus predecesores inmediatos HTML 4.01 y XHTML 1.1, HTML 5 es una respuesta al hecho de que el código HTML y XHTML, de uso común en la web, tienen una mezcla de características introducidas por diversas especificaciones, junto con las introducidas por los productos de software. También es un intento de definir un único lenguaje de etiquetas que puede ser escrito en HTML o XHTML. HTML 5 también es un candidato potencial para aplicaciones móviles multiplataforma. Muchas de las características de HTML5 se han diseñado con dispositivos de baja potencia, tales como teléfonos inteligentes y tabletas (developer.mozilla.org, 2018).

CSS3

Hoja de estilo en cascada (CSS por sus siglas en inglés) parte de un concepto simple pero muy potente: aplicar estilos como colores, formas y márgenes a uno o varios documentos (generalmente documentos HTML, páginas webs) de forma masiva. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas (librosweb.es, 2018).

De esta forma, se puede unificar todo lo relativo al diseño visual en un solo documento CSS, y con ello, varias ventajas:

1. Si necesitamos hacer modificaciones de presentación lo hacemos en un sólo lugar y no tenemos que editar todos los documentos HTML por separado.
2. Se reduce la duplicación de estilos en diferentes lugares, por lo que la información a transmitir es considerablemente menor (las páginas se descargan más rápido).

JavaScript

Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Es un lenguaje interpretado, basado en prototipos que tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios. Es necesario resaltar que hay dos tipos de JavaScript: por un lado, está el que se ejecuta en el cliente, este es el Javascript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un Javascript que se ejecuta en el servidor, es más reciente y se denomina LiveWire JavaScript (Valdés, 2007).

Algunas de las ventajas más destacadas de JavaScript son:

- Es rápido, por lo tanto, tiende a ejecutar las funciones inmediatamente.
- Es soportado por los navegadores más populares y es compatible con los dispositivos más modernos, incluyendo iPhone, móviles y PS3.
- Es multiplataforma, puede ser ejecutado de manera híbrida en cualquier sistema operativo móvil.

1.4.5 Frameworks del lado del cliente

Los frameworks no son nuevos en el desarrollo de aplicaciones de gran envergadura. De hecho, la mayoría de lenguajes consolidados tienen una gran variedad de los mismos. Los frameworks proporcionan la estructura del proyecto y un conjunto de bibliotecas preconfiguradas que ahorran tiempo en el comienzo de los nuevos proyectos. Muchas tareas son delegadas a los frameworks permitiendo a los desarrolladores centrarse exclusivamente en las funcionalidades de su software.

Un marco de trabajo (del inglés frameworks) en lo que refiere a desarrollo web, se trata de un conjunto de procesos, técnicas y archivos previamente confeccionados, que facilitan y aceleran la producción de sitios y aplicaciones web (DesarrolloWeb, 2018).

Para la confección de este trabajo se usará el framework Angular JS por lo siguiente:

Angular JS es un proyecto de código abierto, realizado en JavaScript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo como es el MVC o Modelo Vista Controlador según sus siglas. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso sobre el lenguaje JavaScript con programación del lado del cliente. Este JavaScript pretende que los programadores mejoren su código HTML, ya que se puede producir un HTML altamente semántico, es decir, que cuando sea leído se entienda de manera clara qué es lo que hace o para qué sirve cada cosa. Lógicamente, AngularJS viene cargado con todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear un HTML enriquecido (DesarrolloWeb, 2018).

1.4.6 Tecnologías del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. (Castillo, 2014).

Node.js v.8.12.0

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript. Es una librería y entorno de ejecución de E/S dirigida por eventos y por lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google V8 (LLamas, 2018).

Algunas ventajas de Node.js (LLamas, 2018).

- Se trata de un lenguaje que puede utilizarse en la gran mayoría de servidores, incluyendo los más conocidos como Unix, Microsoft o Mac.
- Garantiza un elevado rendimiento. No sólo puede generar arquitecturas sólidas y potentes, sino que además reduce de una forma muy drástica la ratio de errores.
- Al estar inspirado en JavaScript cuenta con una semántica muy fácil de digerir, aprender y aplicar por cualquier programador.

- Es quizá la opción más competitiva para diseñar aplicaciones que gestionen grandes cantidades de información generadas por una comunidad elevada de usuarios. Un buen ejemplo sería Facebook, una plataforma en la que se generan cientos de miles de comentarios y contenidos por cada segundo.

1.4.7 Sistema Gestor de Base de Datos

PostgreSQL v.9.4

PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas (postgresql.org, 2019).

PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características sólidas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer constantemente soluciones innovadoras y de alto rendimiento. PostgreSQL se ejecuta en todos los sistemas operativos principales, ha sido compatible con ACID desde 2001, y tiene complementos poderosos como el popular extensor de base de datos geoespacial PostGIS. No es sorprendente que PostgreSQL se haya convertido en la base de datos relacional de código abierto elegida por muchas personas y organizaciones (postgresql.org, 2019).

Principales características de este gestor de bases de datos:

- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Estabilidad.
- Alto rendimiento.

1.4.8 Entorno integrado de desarrollo (IDE)

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Microsoft Visual Studio Code v.1.32

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de

código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito, de código abierto y es un editor de código que podamos utilizarlo con los lenguajes de programación que trabajamos a diario. Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros. (Luca, 2016).

Una de ventaja interesante de este editor de código es la posibilidad de configurar la vista a nuestro gusto. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas de nuestro proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades (Luca, 2016).

1.4.9 Herramientas para las pruebas

Jasmine

Jasmine es un marco de desarrollo basado en el comportamiento para probar el código JavaScript. No depende de ningún otro framework de JavaScript. No requiere un DOM. Y tiene una sintaxis limpia y obvia para que pueda escribir pruebas fácilmente(Hahn, 2013).

En Jasmine encontraremos las siguientes características:

- Sintaxis natural para poder probar comportamientos con BDD.
- Soporte para pruebas asincrónicas.
- Personalización de diferentes componentes como reporters y matchers.

Karma.js

Karma es un corredor de prueba para JavaScript que se ejecuta en Node.js. Está muy bien adaptado para probar AngularJS o cualquier otro proyecto de JavaScript. Usar Karma para ejecutar pruebas utilizando una de las muchas suites de pruebas de JavaScript populares (Jasmine, Mocha, QUnit, etc.) y ejecutar esas pruebas no solo en los navegadores de su elección, sino también en la plataforma de su elección (escritorio, teléfono, tableta.) Karma es altamente configurable, se integra con los populares paquetes de integración continua (Jenkins, Travis y Semaphore) y tiene un excelente soporte de complementos (Peter , y otros, 2013).

El objetivo principal de Karma es brindar un entorno de prueba productivo a los desarrolladores. El entorno es uno donde no tienen que configurar un montón de configuraciones, sino un lugar donde los desarrolladores pueden simplemente escribir el código y obtener retroalimentación instantánea de sus pruebas. Porque obtener retroalimentación rápida es lo que te hace productivo y creativo (Peter , y otros, 2013).

1.5 Conclusiones del capítulo

Una vez finalizado este capítulo se arribaron a las siguientes conclusiones:

- Se definieron los conceptos relacionados con el dominio del problema contribuyendo a una mayor comprensión del entorno del negocio.
- Después de realizar una valoración acerca de algunos sistemas de reseña, calificación de servicios y contratación de reparaciones domésticas en el ámbito nacional e internacional, se decidió no emplear ninguno de ellos porque son sistemas privativos que no se adaptan a los requerimientos a los que aspira la empresa de Tecnologías de la Información para la Defensa(XETID).
- Se determinaron a Angular y Node.js como tecnologías para el desarrollo del módulo, así como la utilización de las herramientas Microsoft Visual Studio Code, PostgreSQL, Visual Paradigm, Karma y Jasmine para llevar a cabo dicho proceso. Además, se determinó XP como metodología a seguir para la elaboración de una propuesta de solución que cumpla con el objetivo de la investigación.

CAPITULO 2: Características, análisis y diseño del sistema:

En el presente capítulo se presenta una propuesta de solución para los problemas detectados en el estudio del estado del arte realizado en el capítulo anterior. También se definen las HU, la planificación de entrega de versiones del producto y el diseño del sistema de acuerdo a las fases que propone la metodología seleccionada.

2.1 Modelo de negocio

El modelo de negocio del módulo se realiza a través de un modelo de procesos. Para su modelación se utiliza el diagrama de proceso de negocio para la gestión de reparaciones domésticas.

La plataforma D´Prisa cuenta con un conjunto de software autónomos llamados micro-servicios, probados y funcionales. Dicha plataforma es actualizada y enriquecida constantemente y como parte de este proceso incorpora el módulo que permite la gestión y contratación de servicios de reparaciones domésticas facilitando la contratación de servicios de este tipo. Como primer paso el cliente debe entrar el usuario y contraseña previamente registrados por lo que se hace necesario ser usuario del sistema. A continuación, se accede a la página principal que cuenta con una serie de opciones como Mis ofertas y Servicios. En dependencia de su selección el cliente gestiona sus propias ofertas al poder crear, modificar y eliminar cada una de ellas mientras visualiza los diferentes estados por los que pasan. Siguiendo estos estados el cliente es capaz de contratar y calificar al proveedor que considere indicado para llevar a cabo el trabajo. El cliente puede consultar cada uno de los servicios y solicitar los mismos para su posterior consumición una vez contactado el proveedor que atenderá su solicitud. Además, el módulo permite la interrelación entre el cliente y el proveedor mediante el intercambio de mensajes.

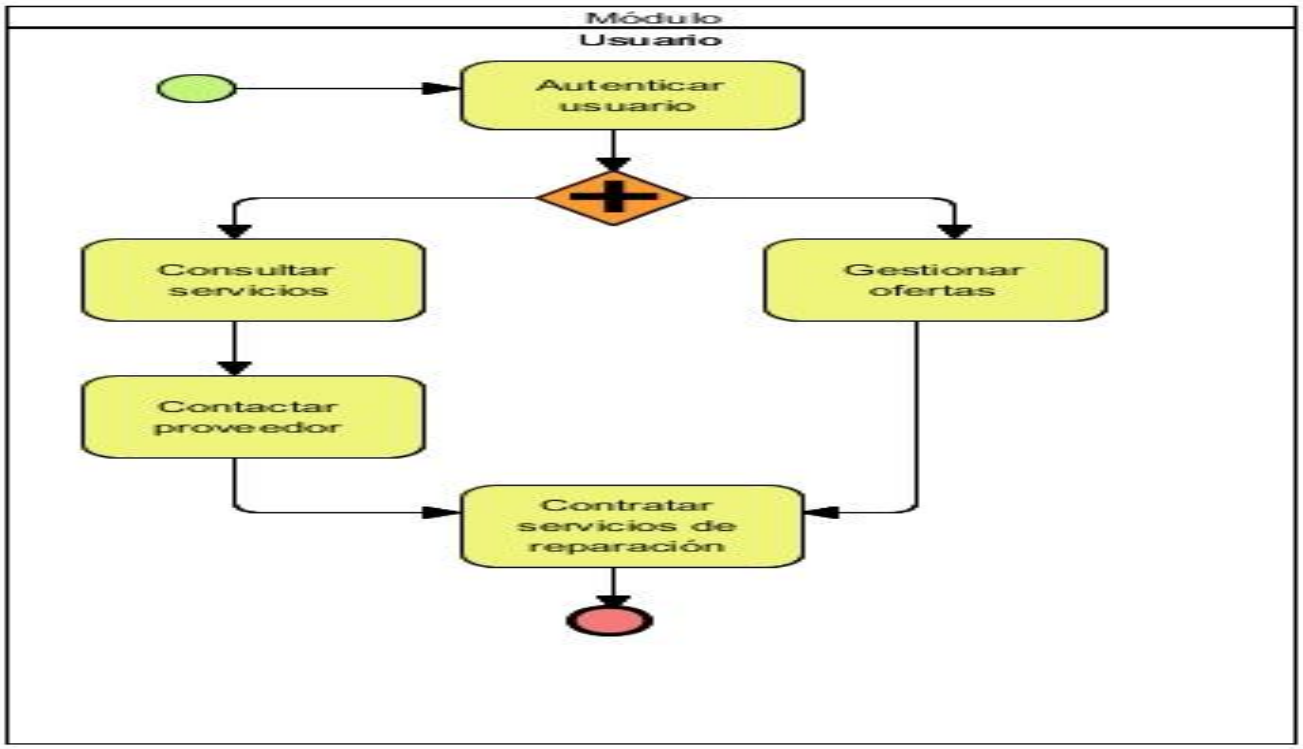


Fig. 6: Modelo de negocio del Módulo Reparaciones Domésticas

2.2 Propuesta de solución

Para dar solución a la problemática planteada se propone la realización de un módulo de clientes para la gestión reparaciones domésticas que responderá a la necesidad de contar con un sistema especializado en la gestión y contratación de dichos servicios, el cual consiste en una aplicación web cuya implementación proporcionará a la empresa XETID la oportunidad de incluir en una de sus plataformas de desarrollo un sistema que permita reseñar y clasificar servicios necesarios en el hogar mediante la contratación del trabajador por cuenta propia que más se ajuste a las necesidades de cada cliente.

El módulo a desarrollar contribuirá en gran medida a que cada trabajador por cuenta propia del sector de reparaciones domésticas que sea usuario del sistema, mediante su previo registro en el mismo, pueda difundir su trabajo y a la vez buscar nuevas ofertas en el mismo, el sistema fomentará la interrelación cliente-proveedor creando un estrecho marco de trabajo, además facilitará el trabajo al cliente necesitado a la hora localizar el proveedor correcto que se ajuste a las necesidades de su reparación, permitiéndole esto un ahorro considerable de tiempo y dinero, todo ello mediante la creación de sus propias ofertas de trabajo, así como la solicitud de los servicios de reparación que necesite su hogar.

Contará con disimiles funcionalidades como la creación y modificación de oferta, solicitud de servicios y contratación del personal calificado que posibilitaran el aumento de la agilidad y la organización de este proceso.

La solución será desarrollada teniendo en cuenta las reglas de negocio definida por la empresa XETID, así como con tecnologías actualizadas de código abierto y libres de licencia.

2.2 Especificación de requisitos de software

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Los requisitos reflejan la necesidad de los clientes de un sistema que ayuda a resolver problemas como el control de un dispositivo, hacer un pedido o encontrar información. El proceso que permite descubrir, analizar, documentar y verificar esos servicios y restricciones se denomina ingeniería de requisitos (SOMMERVILLE, 2005).

2.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (SOMMERVILLE, 2005). El levantamiento de requisitos para el módulo propuesto arrojó como requisitos funcionales los siguientes:

Tabla 2: Requisitos funcionales

No	Requisito Funcional
RF1	Autenticar usuario
RF2	Crear oferta
RF3	Modificar oferta
RF4	Listar ofertas
RF5	Cambiar estado de la oferta
RF6	Eliminar oferta
RF7	Listar proveedores interesados
RF8	Contratar proveedor
RF9	Calificar proveedor

RF10	Listar servicios
RF11	Filtrar servicios
RF12	Listar proveedores por servicios
RF13	Listar proveedores contactados
RF14	Establecer servicio de mensajes

2.2.2. Características no funcionales

Los requisitos no funcionales definen las restricciones del sistema, son propiedades que el sistema debe poseer. Representan las características del producto (SOMMERVILLE, 2005). En el caso del Módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas son las siguientes:

Tabla 3: Características no funcionales.

Identificador	**Descripción**
Usabilidad	
RNF#1	El módulo debe presentar un acceso fácil y rápido.
RNF# 2	Ofrecer interfaz amigable, interactiva e intuitiva.
RNF# 3	El módulo debe permitir el intercambio de mensajes entre diferentes usuarios(roles) una vez integrado(cliente-proveedor).
RNF# 4	El módulo será distribuido en idioma español.
Portabilidad	
RNF# 5	El sistema debe ser multi-plataforma.
Fiabilidad	
RNF# 6	El acceso a la aplicación se realizará mediante roles.
Disponibilidad	
RNF# 7	El servicio web deberá estar disponible las 24 horas del día, asegurando el acceso desde cualquier lugar.
Interfaz	

RNF# 9	La interfaz del módulo contendrá los datos de forma estructurada, permitiendo la interpretación correcta de la información.
RNF# 10	Todos los textos y mensajes en pantalla serán mostrados en idioma español.
RNF# 11	Si ocurre un ocurrir un error con los datos de entrada el módulo muestra mensajes al usuario informando este.
Software	
RNF# 12	Sistema Gestor de Base de Datos PostgreSQL
RNF# 13	Las PC clientes deben tener instalado un navegador web (Mozilla Firefox, Google Chrome, Opera)
Seguridad	
RNF# 14	El módulo debe garantizar la seguridad a través de la autenticación de los usuarios
RNF# 15	Las diferentes áreas del módulo se encontrarán protegidas contra acceso no autorizado utilizando roles y grupos de usuarios.
RNF# 16	La aplicación debe permitir que la contraseña se almacene de manera encriptada en la base de datos.

2.3. Fase I: Planificación

Los procesos de la planificación son la base que sustenta cualquier idea o iniciativa; es decir, dotan de método y estructura a una serie de acciones conjuntas. Cualquier proyecto requiere unos pasos debidamente establecidos que permitan fijar prioridades, definir estrategias y garantizar la toma de decisiones en torno a un objetivo común.

La fase de planificación es la primera fase que propone la metodología XP para el desarrollo de software. En esta fase el cliente define el alcance general que tendrá el proyecto. En la misma explica todas sus necesidades mediante la realización de HU y establece las prioridades para cada una de ellas. Posteriormente los programadores realizan la estimación de tiempo de desarrollo basándose en esta información. Las estimaciones realizadas en esta fase son primarias debido a que están basadas en datos de muy alto nivel y estas podrían variar en posteriores iteraciones.

2.3.1. Historias de Usuarios

Las historias de usuario son breves descripciones del comportamiento de un software. Permiten dividir una gran parte de las funcionalidades en partes más pequeñas para lograr la planificación.

Son escritas en el propio lenguaje del cliente, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. (Patricio Letelier y Mª Carmen Penadés)

A continuación, se muestran las principales historias de usuario generadas durante el desarrollo del módulo Ver Anexo 1.

Tabla 4: Historia de usuario 2.

Historia de usuario	
Número: 2	**Nombre:** Gestionar ofertas
Usuario: Cliente	
Iteración asignada : 1	**Puntos estimados**: 2.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Ofrece al usuario una vista para crear ofertas de trabajo. También permite listar, modificar y eliminar dichas ofertas.	
Observaciones: El usuario podrá visualizar lo proveedores interesados, contactarlos y contratar sus servicios.	

2.3.2 Estimación de esfuerzo por Historia de Usuario

Para un exitoso desarrollo del módulo se propone la siguiente estimación de esfuerzos por cada historia de usuario definida en la fase de exploración. A continuación, se muestran la información requerida:

Tabla 5: Puntos de estimación

Historias de Usuario	Puntos estimados(Semanas)
Autenticar usuario	1.0
Gestionar ofertas	2.0
Mostrar proveedor	2.0
Calificar proveedor	1.0
Mostrar servicios	2.0
Establecer servicio de mensajes	1.0

2.3.3. Desarrollo del plan de iteraciones

Una vez definidas las HU y realizada una previa estimación de esfuerzos, se procede a la planificación de la etapa de implementación del sistema. En este espacio, se crea el plan de iteraciones, donde se especifica la prioridad con que se implementarán las HU organizadas por iteraciones. Teniendo en cuenta el esfuerzo asociado a las mismas y a las prioridades del cliente, se define una versión que sea de valor para este. La planificación que se llevó a cabo para el desarrollo del sistema, está compuesta por dos iteraciones, permitiendo que al final de la última iteración se logre un producto con todas las restricciones y características deseadas por el cliente. (Joskowicz, 2008).

Tabla 6: Duración total de Iteraciones

Iteración	No	Historia de Usuario	Duración de Iteraciones
Iteración #1	1	Autenticar usuario	3 semanas
	2	Gestionar ofertas	
Iteración #2	3	Mostrar proveedor	3 semanas
	4	Calificar proveedor	
Iteración #3	5	Mostrar servicios	3 semanas
	6	Establecer servicio de mensajes	
Total			9 semanas

2.3.4 Plan de entrega

La tabla muestra el plan de entregas, se definen las fechas de inicio y fin de cada historia de usuario basadas en las iteraciones y los días estimados.

Tabla 7: Plan de Entregas

Entregable	**1ra Iteración**	**2da Iteración**	**2da Iteración**
Versión	versión 0.3	versión 0.6	versión 1.0
Fecha	(8-2-2019)	(25-3-2019)	(7-05-2019)

2.4. Fase II: Diseño del sistema

El diseño es fundamental, a diferencia de otras metodologías se realiza durante todo el tiempo de vida del proyecto por lo que se establecen los mecanismos, para que este sea revisado y mejorado continuamente según se van añadiendo funcionalidades al mismo. La metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC, por lo que en esta fase se describen las clases utilizadas en la solución, se analizan las HU y se descomponen en tareas independientes.

2.4.1. Arquitectura de software

La arquitectura de software se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para la guía en el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado (Casanova, 2004).

Para dar a conocer la arquitectura del módulo lo primero que debemos comprender es que cada micro-servicio es un sistema autónomo, por lo que funciona en sí mismo como una aplicación autónoma, exponiendo, independientemente de la arquitectura de la plataforma a la que se integre su propia arquitectura, la cual se describe a continuación:

En el caso de los micro-servicios sigue el patrón de arquitectura MVC (Modelo Vista Controlador) que permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control. Este patrón se ve usualmente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes (Larman, 2000).

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Ejemplo de la estructura del patrón arquitectónico MVC (Modelo Vista Controlador):

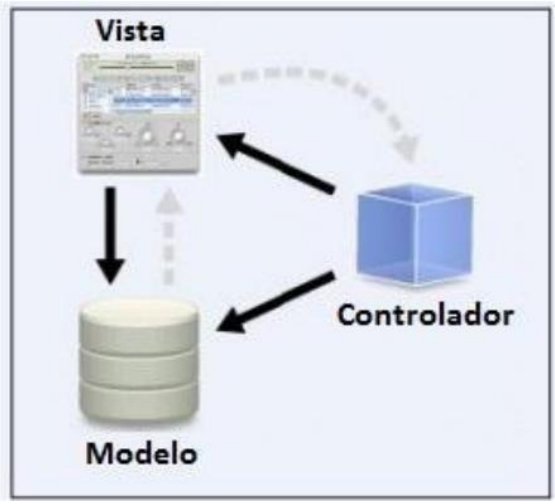


Fig. 7 Patrón arquitectónico MVC(Modelo Vista Controlador)

La principal ventaja de utilizar esta arquitectura es que existe una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio. La separación de capas es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos. Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas.
- Modificar los objetos de negocios para migrar a otra tecnología.
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas.

Luego de haber conocido la arquitectura de los micro-servicios a implementar se pasa a conocer la arquitectura del módulo en general.

Para desarrollo de este módulo se tiene la arquitectura de micro-servicios, conocido por las siglas MSA (Micro Services Architecture), esta es un método de desarrollo de aplicaciones o sistemas de software, compuesto por una colección de servicios autónomos y pequeños, cada mocroservicios es independiente entre sí y cada uno se encarga de implementar una funcionalidad completa del negocio, la comunicación entre estos se realiza a través de API´s (MikeWasson, 2018).

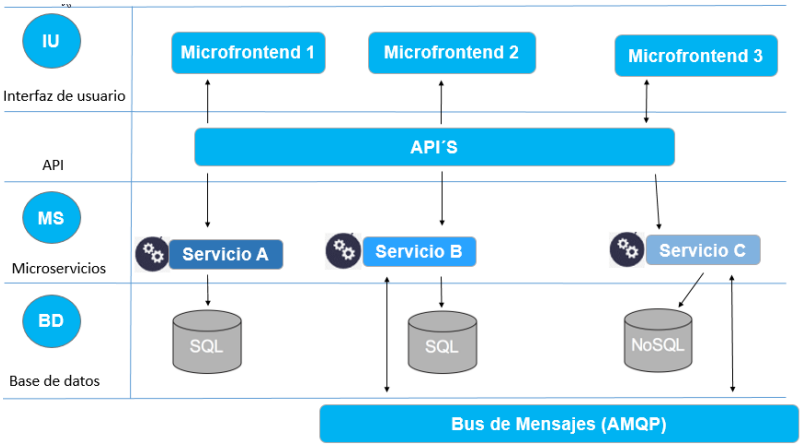


Fig. 8 Arquitectura de micro-servicios de la plataforma D´Prisa.

Como toda arquitectura posee una serie de ventajas y desventajas, a continuación de exponen las principales de cada una:

Ventajas

- **Implementaciones independientes**. Se puede implementar, revertir o poner al día un servicio si algo va mal sin volver a implementar toda la aplicación. Las correcciones de errores y las publicaciones de características son más fáciles de administrar y entrañan menos riesgo.
- **Desarrollo independiente**. Un solo equipo de desarrollo puede compilar, probar e implementar un servicio. El resultado es la innovación continua y un ritmo más rápido de publicación.
- **Aislamiento de errores**. Si un servicio deja de funcionar, no es necesario paralizar toda la aplicación.
- **Pilas de tecnología mixta**. Los micro-servicios permiten el uso de diferentes tecnologías y lenguajes.
- **Reutilización de código.** El desarrollador puede aprovechar las funcionalidades que ya han sido desarrolladas por terceros—no necesita reinventar la rueda, simplemente utilizar lo que ya existe y funciona.

- **Facilidad al desplegar.** Los micro-servicios pueden desplegarse según sea necesario, por lo que funcionan bien dentro de metodologías ágiles.

Desventajas

- **Complejidad**. Una aplicación de micro-servicios tiene más partes en movimiento que la aplicación monolítica equivalente. Cada servicio es más sencillo, pero el sistema como un todo es más complejo.
- **Desarrollo y pruebas**. El desarrollo con dependencias de servicios requiere un enfoque diferente. Las herramientas existentes no están necesariamente diseñadas para trabajar con dependencias de servicios.
- **Falta de gobernanza**. El enfoque descentralizado para la generación de microservicios tiene ventajas, pero también puede causar problemas. Puede acabar con tantos lenguajes y marcos de trabajo diferentes que la aplicación puede ser difícil de mantener.
- **Congestión y latencia de red**. El uso de muchos servicios pequeños y detallados puede dar lugar a más comunicación inter-servicios. Además, si la cadena de dependencias del servicio se hace demasiado larga (el servicio A llama a B, que llama a C), la latencia adicional puede constituir un problema.
- **Integridad de datos**. Cada micro-servicio es responsable de la conservación de sus propios datos. Como consecuencia, la coherencia de los datos puede suponer un problema.

2.4.2 Tarjetas CRC

Teniendo en cuenta el análisis para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). Por tanto, el paso siguiente en el análisis de requerimientos de una aplicación es la creación del modelo CRC.

Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP (Rosado Gómez Alveiro, 2012).

Las principales características de las tarjetas CRC son:

- Identificación de clases y asociaciones que participan del diseño del sistema.
- Obtención de las responsabilidades que debe cumplir cada clase.

- Establecimiento de cómo una clase colabora con otras clases para cumplir con sus responsabilidades.

A continuación, se presentarán algunas de las tarjetas CRC para el módulo a desarrollar Ver Anexo2 :

Tabla 8:Tarjeta CRC #2: Cliente

Tarjeta CRC	
Clase: Cliente	
Responsabilidades:	Colaboraciones:
Se encarga de recoger los datos de los clientes del sistema(nombre, apellidos, usuario, password).	• Servicio

Tabla 9:Tarjeta CRC #3: ClienteController

Tarjeta CRC	
Clase: ClienteController	
Responsabilidades:	Colaboraciones:
registrar (nombre, apellidos, usuario, password): Funcionalidad encargada de crear los usuarios clientes.	Servicio Oferta Calificación

Tabla 10:Tarjeta CRC #6: Calificación

Tarjeta CRC	
Clase: Calificación	
Responsabilidades:	**Colaboraciones:**
Se encarga de almacenar las evaluaciones que los clientes ofrecen por cada servicio recibido.	

Tabla 11:Tarjeta CRC #7: CalificaciónController

Tarjeta CRC	
Clase: CalificaciónController	
Responsabilidades:	**Colaboraciones:**
▪ **getCalificacion**: devuelve la calificación de todos los proveedores del sistema. ▪ **getCalificacionID(**idProveedor**)**: devuelve la calificación de un proveedor del sistema dado su id. ▪ **addCalificacion(**idCliente, idProveedor, calificacion**)**: adiciona la calificación otorgada por una cliente a un proveedor determinado utilizando sus id.	▪ Cliente ▪ Proveedor

Tabla 12:Tarjeta CRC #8: Oferta

Tarjeta CRC	
Clase: Oferta	
Responsabilidades:	**Colaboraciones:**
Se encarga de contener los datos de las ofertas del sistema.	▪ Cliente ▪ Proveedor

Tabla 13:Tarjeta CRC #9: OfertaController

Tarjeta CRC	
Clase: OfertaController	
Responsabilidades:	**Colaboraciones:**

▪ **crearoferta** (tipodeServicio, descripcion): Funcionalidad encargada de crear las ofertas. ▪ **modificar** (tipodeServicio, descripcion): Funcionalidad encargada de modificar una oferta. ▪ **eliminar** (idOferta): Funcionalidad encargada de eliminar una oferta. ▪ **obtOfertas** ()**:** Funcionalidad encargada de obtener todas las ofertas. ▪ **obtOfertaId**(idOferta): Funcionalidad encargada de obtener una oferta dado un id.	▪ Servicio ▪ Provincia ▪ Oferta ▪ Calificación

2.4.3. Selección de patrones

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Gamma, 1995).

En síntesis, son un esqueleto básico que cada diseñador adapta a las peculiaridades de su aplicación. Para el desarrollo del módulo se usó patrones del tipo GRASP (Acrónimo de General Responsibility Assignment Software Patterns) que serán descritos a continuación.

2.4.3.1. Patrones GRASP

Los Patrones GRASP describen los principios fundamentales para asignar responsabilidades a los objetos, existen 3 patrones de este tipo experto, creador y controlador .

Experto: Es un método que promueve la especialización de componentes, lo que significa, agregar responsabilidades a una clase especializada. Es una forma de distribuir la ejecución repartiendo las responsabilidades (Larman, 2004).

Este patrón se pone de manifiesto en cada clase del módulo del sistema, ya que cada una constituye un componente especializado en el manejo de datos asociados a la parte del negocio que les corresponde.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (Larman, 2004).

Este patrón se puede evidenciar en el módulo a desarrollar en las clases controladoras OfertaController y ClienteController a la hora de crear cada uno de los objetos de tipo oferta o cliente.

Alta Cohesión: Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos (Larman, 2004).

Este patrón de evidencia en el diseño del módulo ya que las clases tienen distintas responsabilidades, están estrechamente relacionadas y no realizan un trabajo excesivo, lo que permite simplificar el mantenimiento y aumentar la capacidad de reutilización de estas.

2.5 Conclusiones del capítulo

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- A partir de la definición de las HU se pudieron identificar las principales funcionalidades a desarrollar en correspondencia con el módulo de clientes de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa propuesto.
- La estimación del tiempo para la implementación de las HU definidas, permitió calcular una entrega final del producto en 3 meses aproximadamente.
- El uso de los patrones de diseño contribuyó a la obtención de un diseño con calidad.
- La confección de los artefactos tarjetas CRC permitió dar a conocer las relaciones existentes entre las distintas entidades del módulo.

CAPITULO 3: Implementación y pruebas

En el presente capítulo se muestran las tareas de ingeniería, artefactos generados en la fase de desarrollo de la metodología de software utilizada. Se describen las pruebas efectuadas al software que tienen como objetivo: detectar y corregir el máximo de errores en el sistema, antes de su entrega al cliente.

3.1. Fase III: Desarrollo

En la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar. Las mismas se descomponen en tareas de programación o ingeniería, que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores (Joskowicz, 2008).

3.1.1. Tareas de ingeniería

Tienen como objetivo definir cada una de las actividades que dan cumplimiento a las HU, de forma tal que se entienda lo que el sistema tiene que hacer y facilite su construcción. Pueden estar descritas por un lenguaje técnico y no ser necesariamente entendibles por el cliente. Cada HU como funcionalidad de la aplicación está compuesta por una o varias tareas de ingeniería, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación, se detallan para cada una de las iteraciones las tareas a desarrollar por cada HU (Wallace, 2002).

3.2.2. Desarrollo de las tareas de ingeniería

A continuación, se describen algunas de las tareas definidas Ver Anexo 3 :

Para la primera iteración, se definieron un total de cinco tareas de ingeniería, cada una desglosada a partir de la HU correspondiente.

• HU Gestionar ofertas

Tabla 14. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 2
Nombre de la tarea: Crear oferta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 17 de noviembre de 2018	**Fecha de fin:** 25 de noviembre de 2018

Programador Responsable: Pedro Antonio García González
Descripción: Se implementa una funcionalidad que permite al cliente crear una oferta de trabajo en la que selecciona el tipo de servicio requerido y la descripción de la misma, cada uno de los campos están validados por lo que el usuario no podrá crear la oferta sino llena cada uno de ellos.

Tabla 15. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 2
Nombre de la tarea: Listar ofertas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 26 de noviembre de 2018	**Fecha de fin**: 4 de diciembre de 2018
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todas las ofertas de trabajo creadas y en cada una de ellas el grupo de proveedores que solicitan la oferta y mostrar la calificación de los mismos.	

Tabla 16. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 2
Nombre de la tarea: Modificar oferta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 5 de diciembre de 2018	**Fecha de fin**: 13 de diciembre de 2018
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite al usuario modificar la oferta realizada cambiando tanto el tipo de servicio como la descripción de la oferta.	

Tabla 17. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 2
Nombre de la tarea: Cambiar estado de la oferta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha de inicio: 14 de diciembre de 2018	**Fecha de fin:** 21 de diciembre de 2018
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite chequear cada uno de los estados por los que transita cada oferta desde su creación, posterior aceptación de proveedor que la solicita hasta llegar a la fase terminada.	

Tabla 18. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 2
Nombre de la tarea: Eliminar oferta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 28 enero de 2019	**Fecha de fin:** 8 de febrero de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad, que permite eliminar una oferta previamente seleccionada en fase terminada, eliminando dicha oferta del historial de ofertas terminadas.	

Para la segunda iteración, se definieron un total de cuatro tareas de ingeniería, cada una desglosada a partir de la HU correspondiente. A continuación, se describen algunas de las tareas definidas:

• ***HU Calificar Proveedor***

Tabla 19. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 4
Nombre de la tarea: Calificar proveedor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 6 de marzo de 2019	**Fecha de fin:** 16 marzo de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite al cliente calificar al proveedor que atendió su oferta con un criterio de una a cinco estrellas según la calidad del trabajo realizado, esta funcionalidad es subjetiva a la decisión del cliente de calificar o no al proveedor.	

Para la tercera iteración, se definieron un total de cuatro tareas de ingeniería, cada una desglosada a partir de la HU correspondiente. A continuación, se describen algunas de las tareas definidas:

• ***HU Mostrar servicios***

Tabla 20. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 5
Nombre de la tarea: Listar servicios por categoría	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha de inicio: 17 marzo de 2019	**Fecha de fin:** 30 de marzo de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todos los servicios que se brindan.	

Tabla 21. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 5
Nombre de la tarea: Filtrar servicios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha de inicio: 1 de abril de 2017	**Fecha de fin:** 10 de abril de 2017
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite filtrar una lista con todos los servicios que se brindan, así como los proveedores registrados por cada uno de los servicios dada una provincia seleccionada .	

Con las tareas de ingeniería definidas, se hace necesario establecer un conjunto de pruebas para comprobar la calidad de la solución implementada. Luego, se analizan estos casos de prueba y se ejecutan, lo que permite medir el nivel de cumplimiento con los objetivos de implementación trazados y el nivel de satisfacción del cliente.

3.3. Fase IV: Pruebas

La metodología XP enfatiza mucho los aspectos relacionados con las pruebas ya que anima a probar constantemente o tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección (McBreen, 2002).

En este proceso XP clasifica las pruebas en diferentes tipos y funcionalidades específicas, dividiéndolas en diferentes grupos: pruebas unitarias, desarrolladas por los programadores con el empleo de algún mecanismo que permita automatizarlas de modo tal que tanto su implementación y ejecución consuman el menor tiempo posible permitiendo sacarles el mejor provecho y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Beck, 2000).

Las características clave de las pruebas en XP son (SOMMERVILLE, 2005):

• Desarrollo previamente probado.

• Desarrollo de pruebas incremental a partir de los escenarios.

• Participación del usuario en el desarrollo de las pruebas y en la validación.

3.3.1 Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Las pruebas deben ser definidas antes de realizar el código (López, 2016).

Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (López, 2016).

Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (López, 2016).

Para el módulo se desarrollaron pruebas unitarias a sus métodos mediante la herramienta Karma y Jasmine entre estas podemos encontrar la realizada al método getOfertaID (idOferta) el cual se muestra a continuación:

```
describe('Ofertas', () => {

  const mockResponse =[{"idOferta":54,"tipoServicio":"Carpintería","descripcion":"Poner puerta","idCliente":236,"idProveedor":null,"aceptada":fal
                       {"idOferta":55,"tipoServicio":"Carpintería","descripcion":"Arreglo de ventana","idCliente":236,"idProveedor":null,"aceptad

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [OfertaService]
    });
  });
  describe('Obtener Oferta por id (idOferta) ', () => {
    it('OK',
    inject([HttpTestingController, OfertaService], (httpMock: HttpTestingController, myServiceTested: OfertaService) => {

      myServiceTested.getOfertaId(54)
      .subscribe(
        (res) => {
          expect(res).toEqual(mockResponse);
        }
      );

    })
  );
  });
```

Fig. 9 Método getOfertaID(idOferta)

Método de prueba

Se realizaron pruebas al final de cada iteración contando al concluir con las iteraciones con un total de 11 pruebas. El gráfico que se muestra a continuación muestra los resultados obtenidos en el proceso de pruebas unitarias Ver Anexo 4 .

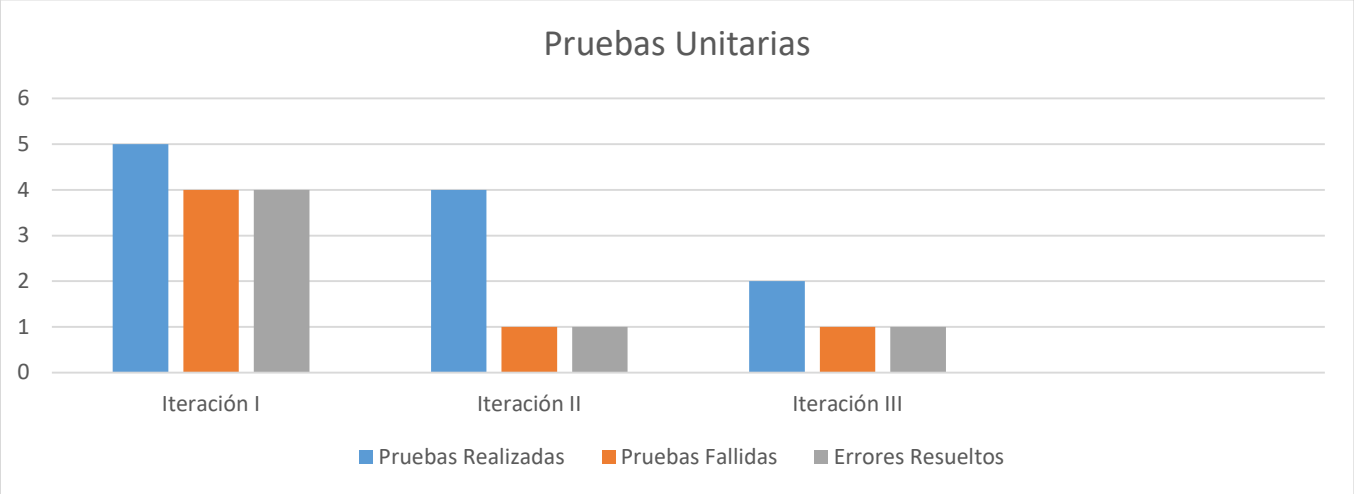


Fig. 10 Resultados de las pruebas unitarias

Como se muestra en la figura anterior se realizaron un total de 11 pruebas unitarias divididas en 3 iteraciones, en la primera de estas iteraciones se realizaron 5 pruebas, de ellas 4 fallaron, por lo que se realizaron 4 correcciones que resolvieron los problemas existentes. En la segunda iteración se realizaron 4 pruebas, develando que existía un fallo en un método el cual fue corregido inmediatamente. Por último, en la tercera iteración se realizaron 2 pruebas donde falló 1 la cual fue corregida de inmediato. Con el cierre de este proceso se comprueba el correcto funcionamiento de las funcionalidades implementadas.

3.3.2. Pruebas de aceptación

Las pruebas de aceptación, también llamadas pruebas funcionales son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario seleccionadas por el cliente deberá tener una o más pruebas de aceptación, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos.

Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia. Es importante resaltar la diferencia entre las pruebas de aceptación y las unitarias en lo que al papel del usuario se refiere. Mientras que en las pruebas de aceptación juega un papel muy importante seleccionando los casos de prueba para cada historia

de usuario e identificando los resultados esperados, en las segundas no tiene ninguna intervención por ser de competencia del equipo de programadores (Luis Miguel Echeverry Tabón, y otros, 2007).

Las pruebas de aceptación tienen más peso que las unitarias ya que constituyen un indicador de la satisfacción del cliente con la solución además de marcar el final de una iteración y el comienzo de la siguiente. Se recomienda que el cliente sea quien diseñe estas pruebas o que al menos participe de manera activa en el proceso.

A continuación, se muestran algunas pruebas de aceptación realizadas para las historias de usuario correspondientes. Para mayor información sobre la descripción de las iteraciones de prueba Ver Anexo 5.

Para la primera iteración, se definieron un total de 5 de pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del módulo.

Tabla 22. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Autenticar usuario.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Autenticar usuario.	
Condiciones de ejecución: El cliente debe de existir en la base de datos.	
Pasos de ejecución: Se llenan los campos según su usuario y su correspondiente contraseña, luego se opta por la opción Acceder. El sistema valida los datos y en caso de ser correctos re-direcciona al cliente a la interfaz principal. En caso que los campos no sea llenados correctamente o el cliente no este registrado se muestra un mensje de alerta de datos incorrectos.	
Resultados esperados: Satisfactorio.	

Tabla 23. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2

Nombre: Crear Oferta .
Descripción: Se comprueba la funcionalidad crear oferta.
Condiciones de ejecución: En el caso de la opción crear oferta, el cliente debe estar autenticado.
Pasos de ejecución: 1. El usuario accede a la interfaz Mis Ofertas. 2. Selecciona la opción Nueva Oferta 3. Completar los campos requeridos con los datos de la oferta. 4. Seleccionar la opción de Agregar. • Si los campos requeridos no se llenados correctamente el sistema no permite la creación de dicha oferta mediante la validación de cada uno de los campos.
Resultados esperados: Satisfactorio.

Para la segunda iteración, se definieron 3 pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del módulo

Tabla 24. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU4_P8	**Historia de usuario:** 4
Nombre: Calificar proveedor.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Calificar proveedor.	
Condiciones de ejecución: El cliente debe de estar autenticado.	
Pasos de ejecución: 1. El usuario accede a la interfaz Mis ofertas. **2.** Una vez creada la oferta se listan los proveedores interesados 3. Se escoge el proveedor que desee. 4. Una vez terminada la oferta se califica al proveedor.	

• Si el cliente no realiza correctamente los pasos antes descritos, el sistema no permite calificar al proveedor seleccionado. Además, si el cliente no desea realizar el proceso de calificación del proveedor solo debe seleccionar la opción cancelar.
Resultados esperados: Satisfactorio.

Para la tercera iteración, se definieron un total de 2 de pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del módulo.

Tabla 25. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU5_P9	**Historia de usuario:** 5
Nombre: Listar servicios.	
Descripción: Se comprueba la funcionalidad Listar servicios.	
Condiciones de ejecución: El cliente debe de estar autenticado.	
Pasos de ejecución: 1. Acceder a la interfaz de servicios. 2. Si se desea filtrar los servicios por algún criterio. 3. Activar la opción de búsqueda avanzada. 4. Escoger el o los filtros que desee.	
Resultados esperados: Satisfactorio.	

Análisis de las pruebas de aceptación

Se desarrollaron un total de catorce casos de pruebas de aceptación. Estas pruebas fueron realizadas de forma organizada, por cada iteración definida. A continuación, se muestra en gráficas, los porcientos de satisfacción alcanzados en cada iteración.

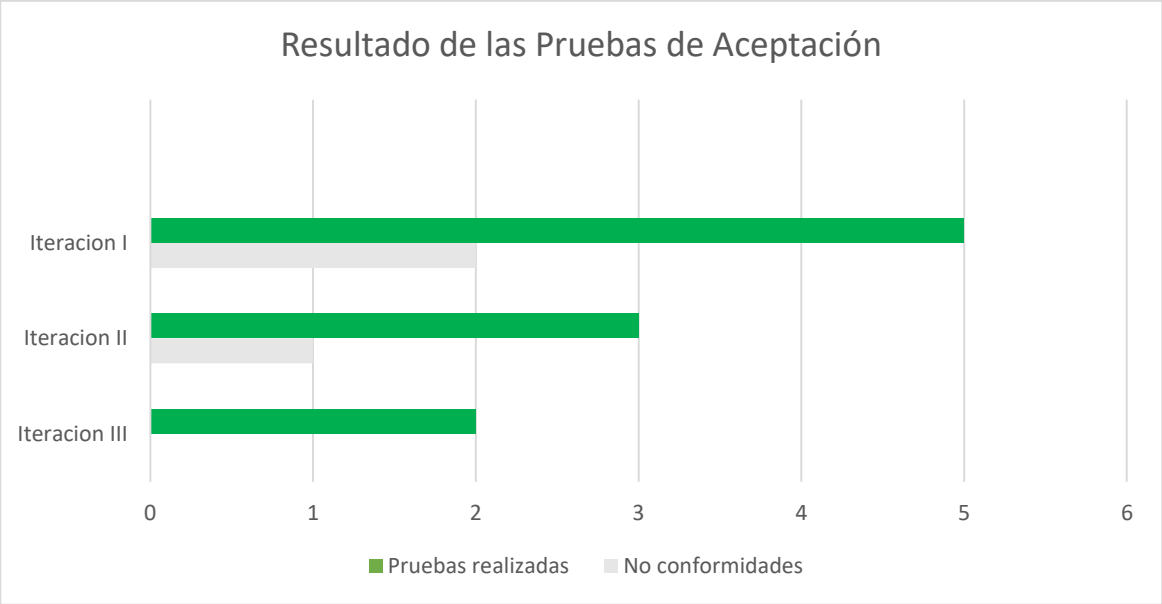


Fig. 11 Resultados de las pruebas de aceptación

Como se puede observar en la figura en la primera iteración se realizaron un total de 5 pruebas, de ellas 3 alcanzaron el nivel de satisfacción esperado, obteniendo un 60% de satisfacción, una de las pruebas, detectó que se creaba la nueva oferta sin validar que los campos estuviesen vacíos. Lo que permitió corregir la falla, e incorporar la seguridad de esta operación en el sistema. En la segunda iteración se realizaron un total de 3 pruebas de ellas 2 alcanzaron el nivel de satisfacción esperado, alcanzando un 67% de satisfacción, una de las pruebas, detectó que una vez listado los proveedores interesados en la oferta, a la hora de contratar a uno de ellos, no se cambia de falso a verdadero el valor del campo aceptada en la base de datos, por lo que se listaban nuevamente todos los proveedores interesados en el estado Ofertas en Ejecución. Lo que permitió corregir el error en el menor tiempo posible. En tanto en la tercera iteración se realizaron 2 pruebas, donde todas son evaluadas de resultado satisfactorio. Con estas comprobaciones, se obtiene un 100% de satisfacción en la iteración final del producto, comprobando el correcto funcionamiento de las funcionalidades implementadas.

3.3.3 Pruebas de Integración

Pruebas de integración es “una técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados a la interfaz” (Pressman, 2005).

Estas son definidas para verificar el correcto ensamblaje entre los distintos módulos que conforman un sistema informático. Las mismas validan que estos componentes realmente funcionan juntos, son llamados

correctamente y, además, transfieren los datos correctos en el tiempo preciso y por las vías de comunicación establecidas (Sommerville, 2005).

Para la aplicación de las pruebas mencionadas anteriormente se utilizan principalmente técnicas que verifican el correcto manejo de las entradas y salidas del software, es decir, las pruebas funcionales. Existen dos tipos de pruebas de integración (Suarez, 2012):

- Incremental
- No incremental

Dentro de las pruebas de integración incremental existen dos enfoques principales como son:

- Integración descendente (componentes funcionales).
- Integración ascendente (componentes de infraestructura).

Para la realización de pruebas de integración del módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas con el sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de micro-servicios sobre la plataforma D'Prisa se eligió el tipo incremental enfocándose en la ascendente la misma permite probar el programa en pequeñas porciones lo que facilita la detección de los errores existentes.

Para probar la integración de los módulos se realizó lo siguiente:

1. Se compiló el frontend del módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa por el puerto 4200(localhost:4200) y el backend por el puerto 3000(localhost:3000).
2. Se compiló el frontend del módulo de clientes para la gestión de reparaciones domésticas para cuentapropistas por el puerto 4201(localhost:4201) y el backend por el puerto 3001(localhost:3001).
3. La base de datos de ambos módulos es la misma y se compiló por el puerto 5432(localhost: 5432)

A continuación, se muestra el caso de prueba de integración calificar proveedor Ver Anexo 6:

Tabla 26:Caso de prueba de integración Calificar proveedor

<table>
<tr><td colspan="3">Caso de prueba de integración: Calificar proveedor</td></tr>
<tr><td colspan="3">Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de micro-servicios sobre la plataforma D’Prisa</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Un cliente debe haber dado por terminada la oferta.</td></tr>
<tr><td colspan="3">Descripción de la prueba: Comprobar que el módulo es de capaz de calificar un proveedor.</td></tr>
<tr><td colspan="3">Entradas/Pasos de ejecución:
1. Se selecciona la opción Terminar Oferta.
2. Marcar calificación a otorgar.
3. Se selecciona la opción Calificar.</td></tr>
<tr><td colspan="3">Resultado esperado: El sistema es capaz de calificar un proveedor.</td></tr>
<tr><td colspan="2">Interfaz Cliente:</td><td>Interfaz Proveedor:</td></tr>
<tr><td colspan="2"> 


</td><td> 
</td></tr>
<tr><td colspan="3">Evaluación: Prueba satisfactoria.</td></tr>
</table>

3.4 Conclusiones del capítulo

Tras haber desarrollado una de las fases más importantes de la metodología XP se arribó a las siguientes conclusiones parciales:

- Las pruebas unitarias realizadas a varios algoritmos del sistema posibilitaron la detección de errores, obteniéndose resultados satisfactorios.

- Las pruebas de aceptación realizadas permitieron comprobar el grado de conformidad del cliente.
- El uso de estándares de codificación permitió generar un cogido legible.
- El plan de entrega fue cumplido acorde al tiempo planificado para el desarrollo de las tareas por cada historia de usuario.

Conclusiones Generales

Tras el desarrollo del módulo de clientes de servicios de reparaciones domésticas empleando la arquitectura de micro-servicios sobre la plataforma D'Prisa, se arriba a las siguientes conclusiones, evidenciando el cumplimiento de los objetivos propuestos:

- Los métodos científicos empleados en la investigación posibilitaron identificar los conceptos que sustentan la presente investigación.
- El estudio de los sistemas homólogos demostró la ausencia de una aplicación que cumpla con las necesidades del centro.
- Aplicando la metodología de desarrollo de software, las herramientas y lenguajes de programación seleccionados, se desarrolló un módulo que cumplió las expectativas del cliente.
- Se aplicaron pruebas de software que posibilitó evaluar el producto, comprobándose el correcto funcionamiento del módulo.

Recomendaciones

A lo largo de la presente investigación fueron identificados elementos que no se contemplaron en el desarrollo de la solución al no ser parte del alcance inicial, pero se considera que son relevantes e incrementarían la utilidad del sistema y la calidad de uso del mismo por lo que se recomienda:

A los futuros desarrolladores que trabajen en la herramienta:

- Implementar la funcionalidad de añadir imagen de la oferta realizada.
- Implementar otras funcionalidades que faciliten el uso de la aplicación tales como: evitar la duplicación de proveedores en la lista de mensajes.

A la XETID:

- Realizar una investigación para determinar nuevas funcionalidades que se puedan agregar al módulo.

Glosario de términos

Tarjetas C.R.C: Clases Responsabilidades Colaboradores. Describe brevemente las responsabilidades de cada clase, así como con las demás clases con que trabaja de forma conjunta.

UML: Lenguaje Unificado de Modelado o UML, por sus siglas en inglés, Unified Modeling Language. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

XP: La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software para el desarrollo de aplicaciones.

IDE: Entorno de Desarrollo Integrado. Es un programa informático que contiene un conjunto de herramientas de programación.

HTML: (Hyper Text Markup Language o Lenguaje de marcado de hipertexto en español) Es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet.

Referencias Bibliográficas

Peter , Darwin Bacon y Kozlowski, Pawel. 2013. *AngularJS web application development.* s.l. : Packt Publ., 2013. - 372 p., 2013. 9781782161820.

2018. Amazon API Gateway. [En línea] 2018. [Citado el: 26 de 11 de 2018.] https://aws.amazon.com/es/api-gateway/.

Amodeo, Enrique. 2012. CQRS (1): ¿Qué es? [En línea] 2012. [Citado el: 21 de 11 de 2018.] https://eamodeorubio.wordpress.com/2012/09/03/cqrs-1-que-es/.

APIs, Drupal 8. 2018. Drupal 8 APIs. [En línea] 2018. https://www.drupal.org/docs/8/api.

Beck, Kent. 2000. *Extreme programming explained: embrace change.* s.l. : Addison-Wesley Professional, 2000.

Beltrán, Pedro. 2012. Qué es una herramienta CASE. [En línea] academia.edu, 23 de 3 de 2012. [Citado el: 16 de 11 de 2018.] http://www.academia.edu/28037284/Qu%C3%A9_es_una_herramienta_CASE.

Blanco, Carlos. 2008. *Patrones de Diseño. Ingeniería del Software I.* s.l. : Universidad de Cantabria, 2008.

Calvo, Diego. 2018. Metodología SCRUM (Metodología ágil). [En línea] 2018. [Citado el: 3 de 12 de 2018.] http://www.diegocalvo.es/metodologia-scrum-metodologia-agil/.

—. 2018. Metodología XP Programación Extrema (Metodología ágil). [En línea] 2018. [Citado el: 3 de 12 de 2018.] http://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/.

Carbó, Yosvany Medina. 2016. Cuba y el impacto de las TIC en la informatización de la sociedad. [En línea] 2016. [Citado el: 15 de 11 de 2018.] https://www.monografias.com/trabajos109/cuba-y-impacto-tic-informatizacion-sociedad/cuba-y-impacto-tic-informatizacion-sociedad.shtml.

Casanova, J. 2004. Usabilidad y arquitectura del software. [En línea] DesarrolloWeb.com, 2004. [Citado el: 21 de 4 de 2019.] http://www.desarrolloweb.com/articulos/1622.php.

Castillo, Erick Jesus Martínez. 2014. Lenguajes de programación del lado servidor. [En línea] 2014. [Citado el: 11 de 12 de 2018.] https://michelletorres.mx/lenguajes-de-programacion-del-lado-servidor/.

Cornejo, José Enrique González. 2008. ¿Qué es UML? El Lenguaje de Modelado Unificado. [En línea] 2008. [Citado el: 5 de 11 de 2018.] https://www.docirs.cl/uml.htm.

Daniel López, Edgar Maya. 2017 . *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.* s.l. : Universidad Técnica del Norte, Instituto de Posgrados, 2017 .

DesarrolloWeb. 2018. Qué es AngularJS. [En línea] DesarrolloWeb.com, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html.

developer.mozilla.org. 2018. Documentación web de MDN. [En línea] developer.mozilla.org, 15 de 11 de 2018. [Citado el: 14 de 11 de 2018.] https://developer.mozilla.org/es/docs/HTML/HTML5.

Diaz, Ivan Alvarado. 2014. ¿Que es FrontEnd Y Backend en la programación web? [En línea] 2014. [Citado el: 24 de 11 de 2018.] https://serprogramador.es/que-es-frontend-y-backend-en-la-programacion-web/.

2018. *Diseño de microservicios: consideraciones de datos.* 2018.

docs.microsoft.com. 2019. Tecnologías web comunes del lado cliente. *Tecnologías web comunes del lado cliente.* [En línea] 2019.

Empresa-3digits. 2018. Microsoft .NET. [En línea] 2018. [Citado el: 10 de 12 de 2018.] https://www.3digits.es/desarrollo/Desarrollo_de_software_y_programacion_de_aplicaciones_web_con_Microsoft_Visual_Studio_NET_Framework.html.

2016. Entorno de Desarrollo Integrado. [En línea] 2016. [Citado el: 12 de 12 de 2018.] https://sites.google.com/site/softwaredeprogramacion2/entorno-de-desarrollo-integrado.

2018. Estrategia Digital Orientada a Resultados, HTML 5. [En línea] 2018. [Citado el: 10 de 12 de 2018.] https://www.arimetrics.com/glosario-digital/html5.

fergarciac. 2013. Entorno de Desarrollo Integrado (IDE). [En línea] 2013, 25 de 1 de 2013. [Citado el: 13 de 11 de 2018.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

2018. FUNDAMENTOS DE INGENIERÍA DE SOFTWARE. [En línea] 2018. [Citado el: 7 de 12 de 2018.] https://sites.google.com/site/ingenierialeosw/unidad-1-fundamentos-de-ingenieria-de-software/1-5-definicion-e-historia-de-las-herramientas-case.

Gamma, Erich. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Pearson Education, 1995.

—. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software. s.l.* s.l. : Pearson Education, 1995.

Geers, Michael. 2018. What are Micro Frontends? [En línea] 2018. [Citado el: 23 de 11 de 2018.] https://micro-frontends.org/.

Henao, Jose Mario Valencia. 2018. Diseño y aplicación de un sistema bajo arquitectura cliente servidor para la activación de riego automatizado en Arduino a traves de Twitter. [En línea] 2018. [Citado el: 20 de 11 de 2018.] https://www.researchgate.net/publication/328174883_Diseno_y_aplicacion_de_un_sistema_bajo_arquitectura_cliente_servidor_para_la_activacion_de_riego_automatizado_en_Arduino_a_traves_de_Twitter.

Iyair Armando Díaz Sánchez, Universidad Politécnica de Puebla. 2011. ¿Que es UML?, diseñodesistemas_iads. [En línea] 2011. [Citado el: 5 de 12 de 2018.] https://sites.google.com/site/disenodesistemasiads/-inicio.

Joskowicz, Ing. José. 2008. *Reglas y Prácticas en eXtreme Programming.* 2008.

Larman, Craig. 2004. *UML y Patrones.* 2004.

—. 2000. *UML y Patrones, 2da Edición.* 2000.

2018. Lenguaje CSS. [En línea] 2018. [Citado el: 10 de 12 de 2018.] https://lenguajecss.com/p/css/introduccion/que-es-css.

librosweb.es. 2018. Capítulo 1. Introducción (Introducción a CSS). [En línea] librosweb.es, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://librosweb.es/libro/css/capitulo_1.html.

LLamas, Luis. 2018. Ingeniería, informática y diseño. *QUÉ ES NODE.JS Y PORQUE YA DEBERÍAS ESTAR USÁNDOLO.* [En línea] 15 de 11 de 2018. [Citado el: 15 de 11 de 2018.] http://noticias.universia.es/ciencia-tecnologia/noticia/2017/07/07/1154054/nodejs-sirve.html.

López, Juan Miguel Bonilla. 2017. *¿Módulos o microservicios?* 2017.

López, Yolanda Borja. 2016. *Metodología Ágil de Desarrollo de Software – XP.* s.l. : ESPE, MEVAST, 2016.

Luca, Damián De. 2016. Desarrollo Web Capacitación y Consultoría. *Desarrollo Web Capacitación y Consultoría.* [En línea] 2016. https://damiandeluca.com.ar/visual-studio-code-caracteristicas-principales.

Luis Miguel Echeverry Tabón y Delgado Carmona, Luz Elena . 2007. *Caso Prático de la metodología ágil xp al desarrollo de software.* s.l. : Universiada Tecnológica de Pereira, 2007.

Mariñán, Martín Pérez. *Patrones de Diseño (Design Patterns).*

McBreen, Pete. 2002. *QuestioningExtremeProgramming.* s.l. : Addison-WesleyProfessional, 2002.

2018. MDN web docs, Frameworks Web de lado servidor. [En línea] 2018. [Citado el: 12 de 12 de 2018.] https://developer.mozilla.org/es/docs/Learn/Server-side/Primeros_pasos/Web_frameworks.

MikeWasson. 2018. Microservices architecture style. *Azure Application Architecture Guide.* [En línea] 2018. https://github.com/MicrosoftDocs/architecture-center/blob/master/docs/guide/architecture-styles/microservices.md.

Oterino, Ana M. del Carmen García. 2015. ¿Qué es eso de los microservicios? [En línea] 2015. [Citado el: 20 de 11 de 2018.] http://www.javiergarzas.com/2015/06/microservicios.html.

Patricio Letelier y Mª Carmen Penadés, Universidad Politécnica de Valencia. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

postgresql.org. 2019. PostgreSQL. [En línea] postgresql.org, 2019. [Citado el: 13 de 3 de 2019.] https://www.postgresql.org/about/.

Pressman, Roger. 2005. *Ingeniería del Software: Un Enfoque Práctico. Sexta.* México : MCGRAW-HILL, 2005. 9789701054734.

Rosado Gómez Alveiro, Quintero Duarte Alexander y Meneses Guevara Cesar Daniel. 2012. 2012.

Saint-Paul, Arnaud. 2008. La información es fuente de poder para el individuo. [En línea] 2008. [Citado el: 15 de 11 de 2018.] https://www.tendencias21.net/La-informacion-es-fuente-de-poder-para-el-individuo_a2038.html.

Santiago, Joaquín Quintas. 2018. La Habana : s.n., 2018.

SOMMERVILLE, IAN. 2005. *Ingeniería del software.* Madrid : Miguel Martín-Romo. PEARSON EDUCACIÓN, SA, 2005.

—. 2005. *Ingeniería del software.* Madrid : Miguel Martín-Romo. PEARSON EDUCACIÓN., 2005.

Sommerville, Ian. 2005. *Ingeniería del Software. Septima edición.* Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

Suarez, Danays Rodríguez. 2012. *Importancia de las pruebas de integración en el control de la calidad de.* Cuba : s.n., 2012.

Valdés, Damián Pérez. 2007. *Maestros de la web, ¿Qué es Javascript?* 2007.

Vidal, Wilfredo González. 2014. Las TICs tienen potencial para el desarrollo de nuestros países. [En línea] 2014. [Citado el: 16 de 11 de 2018.] http://www.cubadebate.cu/opinion/2014/04/25/las-tics-tienen-potencial-para-el-desarrollo-de-nuestros-paises/.

Villegas, Adrian Anaya. 2015. monografias. [En línea] 2015. [Citado el: 30 de 11 de 2018.] https://www.monografias.com/trabajos47/desarrollo-software/desarrollo-software.shtml.

2018. Visual Paradigm. [En línea] 2018. [Citado el: 7 de 12 de 2018.] https://www.visual-paradigm.com/.

Visual Paradigm. 2009. Visual Paradigm. [En línea] Visual Paradigm, 21 de 2 de 2009. [Citado el: 15 de 11 de 2018.] http://www.visual-paradigm.com/.

Wallace, Doug y Ragget. 2002. *Extreme Programming for Web Projects.* s.l. : Addison Wesley, 2002. 0-201-79427-6.

Anexos

A.1 Historias de Usuarios

Tabla 27: Historia de usuario 1

Historia de usuario	
Número: 1	**Nombre:** Autenticar usuario
Usuario: Cliente	
Iteración asignada : 1	**Puntos estimados**: 1.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Luego de que el usuario escoja la opción de acceder el sistema mostrara la interfaz correspondiente, en la cual el usuario tendrá que introducir sus datos.	
Observaciones: Si esta errónea la contraseña o el usuario el sistema muestra un mensaje de alerta.	

Tabla 28: Historia de usuario 3

Historia de usuario	
Número: 3	**Nombre:** Mostrar proveedores
Usuario: Cliente	
Iteración asignada : 2	**Puntos estimados**: 2.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Ofrece al usuario una vista con la lista de proveedores que solicitan cada oferta de trabajo. Además ofrece la posibilidad de contratar el proveedor de su interés.	
Observaciones: Una vez seleccionado el proveedor este se muestra en el estado de Oferta en Ejecución.	

Tabla 29: Historia de usuario 4

Historia de usuario	
Número: 4	**Nombre:** Calificar proveedor
Usuario: Cliente	

Iteración asignada : 2	**Puntos estimados**: 1.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Ofrece al usuario una vista con el proveedor que llevo a cabo la oferta para su calificación.	
Observaciones: El usuario podrá calificar de una a cinco estrellas el proveedor que llevo a cabo su oferta	

Tabla 30: Historia de usuario 5

Historia de usuario	
Número: 5	**Nombre:** Mostrar servicios
Usuario: Cliente	
Iteración asignada : 3	**Puntos estimados**: 2.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Ofrece una vista que permite al usuario observar los servicios que puede solicitar, además pude buscar los diferentes servicios por la provincia deseada.	
Observaciones: Una vez seleccionado el servicio deseado se muestra la lista de posibles proveedores a contratar que una vez contactados se añade a la lista de contactos.	

Tabla 31: Historia de usuario 6

Historia de usuario	
Número: 5	**Nombre:** Establecer servicio de mensajes
Usuario: Cliente	
Iteración asignada : 2	**Puntos estimados**: 1.0
Prioridad: Alta	**Complejidad**: Alto
Descripción: Ofrece al usuario una vista para enviar mensajes a los proveedores interesados en su oferta.	
Observaciones:	

A.2 Tarjetas CRC

Tabla 32:Tarjeta CRC #1 Proveedor

Tarjeta CRC	
Clase: Proveedor	
Responsabilidades:	Colaboraciones:
Contener los datos de los proveedores del sistema. • **getCalificacion**(idProveedor): devuelve la calificación de todos los proveedores del sistema. • **getProveedor**(): devuelve todos los proveedores del sistema. • **getProveedorID** (idProveedor): devuelve un proveedor del sistema dado un id.	

Tabla 33:Tarjeta CRC #4: Servicio

Tarjeta CRC	
Clase: Servicio	
Responsabilidades:	Colaboraciones:
Es la encargada de contener todos los servicios que se podrían prestar los proveedores.	

Tabla 34:Tarjeta CRC #5: ServicioController

Tarjeta CRC	
Clase: ServicioController	
Responsabilidades:	**Colaboraciones:**

▪ **getServicio** (idServicio): encargada de obtener un servicio por su id. ▪ **getServicio**( ): encargada de obtener todos los servicios.	▪ Proveedor ▪ Cliente

Tabla 35:Tarjeta CRC #10: Provincia

Tarjeta CRC	
Clase: Provincia	
Responsabilidades:	**Colaboraciones:**
Se encarga de contener cada provincia del sistema	

Tabla 36:Tarjeta CRC #11: Municipio

Tarjeta CRC	
Clase: Municipio	
Responsabilidades:	**Colaboraciones:**
Se encarga de contener los municipios de cada provincia del sistema	

A.3 Desarrollo de las tareas de ingeniería

• ***HU Autenticar Usuario***

Tabla 37. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha de inicio: 9 de noviembre de 2018	**Fecha de fin:** 16 de noviembre de 2018
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite al usuario previamente registrado acceder al módulo.	

• ***HU Mostrar proveedor***

Tabla 38. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 3
Nombre de la tarea: Listar proveedores interesados	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 9 de febrero de 2019	**Fecha de fin:** 22 de febrero de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todos los proveedores que están interesados en la oferta realizada.	

Tabla 39. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 3
Nombre de la tarea: Contratar proveedor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 23 de febrero de 2019	**Fecha de fin:** 5 de marzo de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite al usuario contratar el proveedor que desee para llevar su oferta	

• ***HU Mostrar servicios***

Tabla 40. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 5
Nombre de la tarea: Listar proveedores por servicios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha de inicio: 11 de abril de 2019	**Fecha de fin:** 19 de abril de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todos los proveedores que se pueden contactar dado el servicio seleccionado.	

• ***HU Establecer servicio de mensajes***

Tabla 41. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 5
Nombre de la tarea: Listar proveedores contactados	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 20 de abril de 2019	**Fecha de fin:** 26 de abril de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todos los proveedores que se han contactados llenando una lista de contactos.	

Tabla 42. Tarea de ingeniería # 14

Tarea	
Número de tarea: 14	**Número de Historia de usuario:** 6

Nombre de la tarea: Establecer servicio de mensajes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha de inicio: 27 de abril de 2019	**Fecha de fin:** 7 de mayo de 2019
Programador Responsable: Pedro Antonio García González	
Descripción: Se implementa una funcionalidad que permite al cliente enviar mensajes al proveedor interesado en su oferta.	

A.4 Pruebas Unitarias

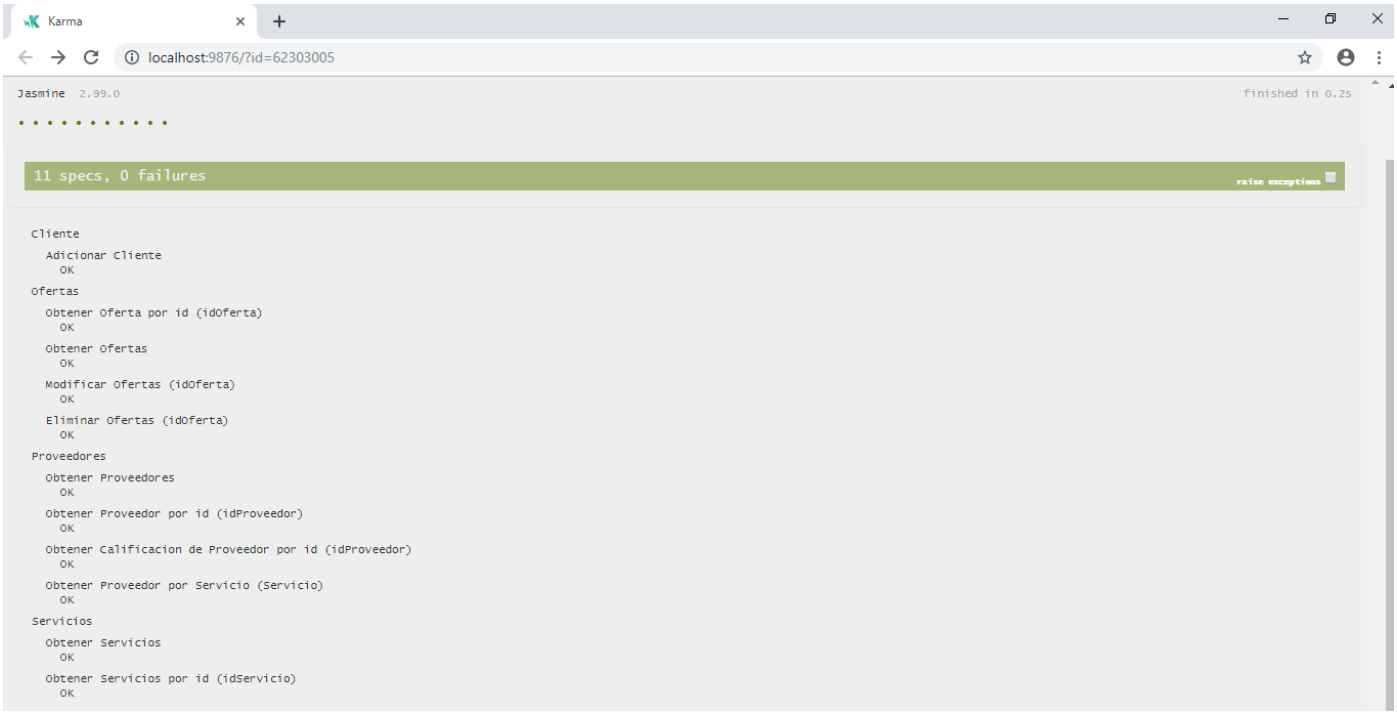


Fig. 12 Pruebas Unitarias

A.5 Pruebas de aceptación

Tabla 43. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU2_P3	**Historia de usuario:** 2

Nombre: Listar Oferta .
Descripción: Se comprueba la funcionalidad crear oferta.
Condiciones de ejecución: En el caso de la opción crear oferta, el cliente debe estar autenticado.
Pasos de ejecución: 1. El usuario accede a la interfaz Mis Ofertas 2. Selecciona la opción Ofertas Pendientes
Resultados esperados: Satisfactorio.

Tabla 44. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU2_P4	**Historia de usuario:** 2
Nombre: Modificar Oferta .	
Descripción: Se comprueba la funcionalidad modificar oferta.	
Condiciones de ejecución: En el caso de la opción modificar oferta, esta debe de existir en la base de datos y estar autenticado el cliente correspondiente.	
Pasos de ejecución: 1. El usuario accede a la interfaz Mis Ofertas. 2. Selecciona la opción modificar en el estado Ofertas Pendientes. 3. Modificar el o los campos requeridos con los datos de la oferta. 4. Seleccionar la opción de Guardar. • Si no se realiza correctamente los pasos antes señalados la oferta se guarda con los datos que tenía antes de seleccionar la opción modificar.	
Resultados esperados: Satisfactorio.	

Tabla 45. Prueba de aceptación # 5

Caso de prueba de aceptación

Código: HU2_P5	**Historia de usuario:** 2
Nombre: Eliminar Oferta .	
Descripción: Se comprueba la funcionalidad eliminar oferta.	
Condiciones de ejecución: En el caso de la opción eliminar oferta, esta debe de existir en la base de datos y estar autenticado el cliente correspondiente.	
Pasos de ejecución: 1. Acceder a la interfaz de ofertas terminadas. 2. Seleccionar la opción de Eliminar Oferta.	
Resultados esperados: Satisfactorio.	

Tabla 46. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU3_P6	**Historia de usuario:** 3
Nombre: Listar proveedores interesados.	
Descripción: Se comprueba la funcionalidad listar proveedores interesados.	
Condiciones de ejecución: El cliente debe de estar autenticado.	
Pasos de ejecución: 1. El usuario accede a la interfaz de Mis Ofertas. 2. Al seleccionar el estado Ofertas Pendientes se muestran los proveedores interesados en dicha oferta.	
Resultados esperados: Satisfactorio.	

Tabla 47. Prueba de aceptación # 7

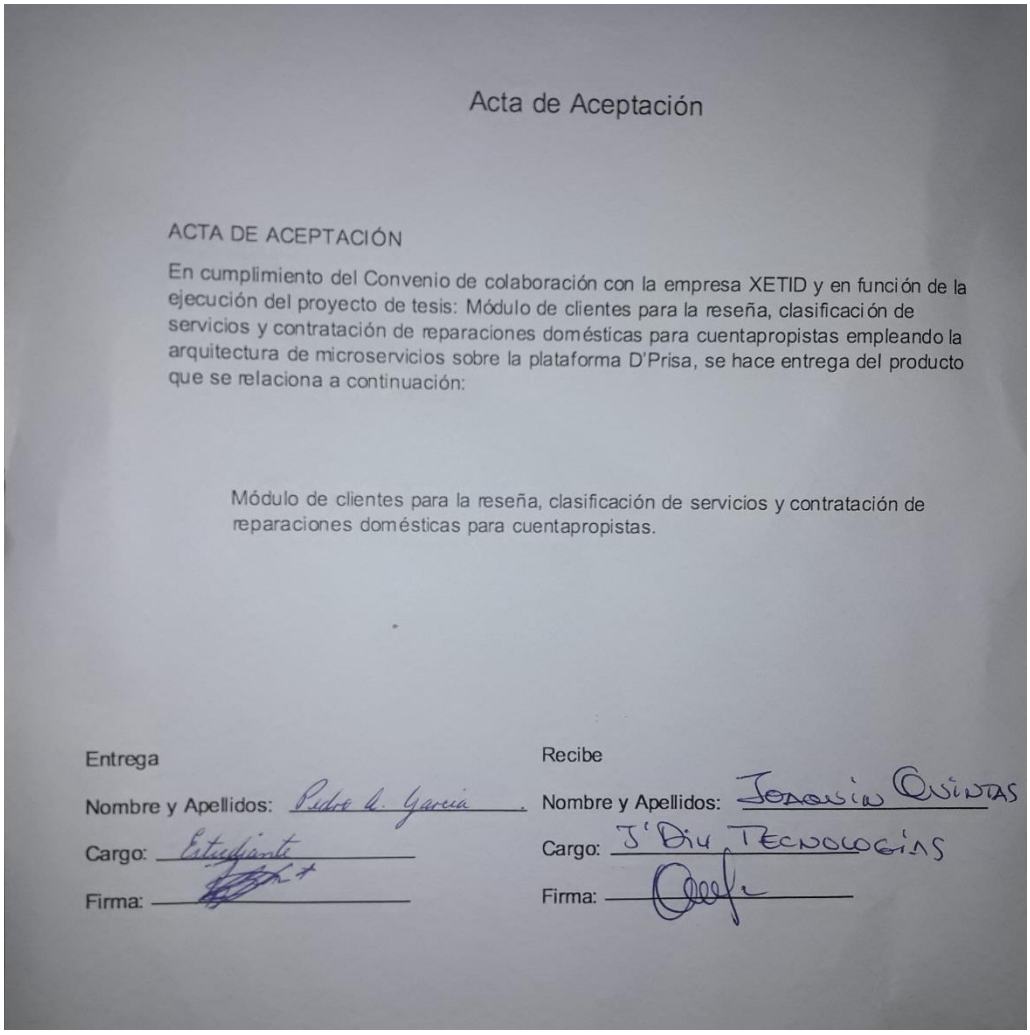
Caso de prueba de aceptación	
Código: HU3_P7	**Historia de usuario:** 3

Nombre: Contratar proveedor.
Descripción: Se comprueba la funcionalidad listar contratar proveedor.
Condiciones de ejecución: El cliente debe de estar autenticado.
Pasos de ejecución: 1. El usuario accede a la interfaz de Mis Ofertas. 2. Al seleccionar el estado Ofertas Pendientes se muestran los proveedores interesados en dicha oferta. 3. Se selecciona lo opción contratar en el proveedor de su interés.
Resultados esperados: Satisfactorio.

Tabla 48. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU6_P10	**Historia de usuario:** 5
Nombre: Establecer servicio de mensajes.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Establecer servicio de mensajes.	
Condiciones de ejecución: El cliente debe de estar autenticado y debe haber contratado algún proveedor para una oferta creada previamente .	
Pasos de ejecución: 1. Acceder a la interfaz Mis ofertas. 2. Escoger al proveedor deseado. 3. Enviar mensajes al proveedor .	
Resultados esperados: Satisfactorio.	

A.5 Certificado de aceptación del producto



Acta de Aceptación

ACTA DE ACEPTACIÓN

En cumplimiento del Convenio de colaboración con la empresa XETID y en función de la ejecución del proyecto de tesis: Módulo de clientes para la reseña, clasificación de servicios y contratación de reparaciones domésticas para cuentapropistas empleando la arquitectura de microservicios sobre la plataforma D'Prisa, se hace entrega del producto que se relaciona a continuación:

Módulo de clientes para la reseña, clasificación de servicios y contratación de reparaciones domésticas para cuentapropistas.

Entrega

Nombre y Apellidos: Pedro A. García

Cargo: Estudiante

Firma:

Recibe

Nombre y Apellidos: Joaquín Quintas

Cargo: J'Diu Tecnologías

Firma:

Fig. 13 Certificado de aceptación del producto

A.6 Pruebas de integración

Tabla 49: Caso de prueba de integración Recibir solicitud

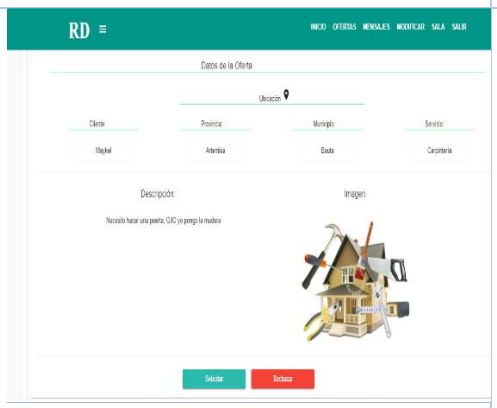
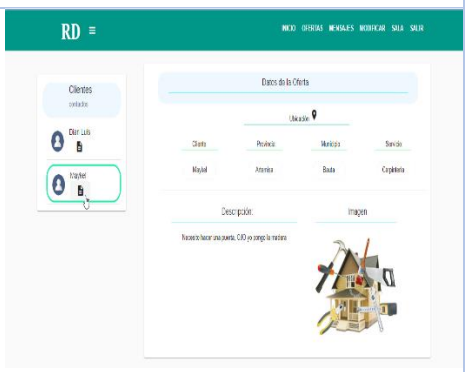
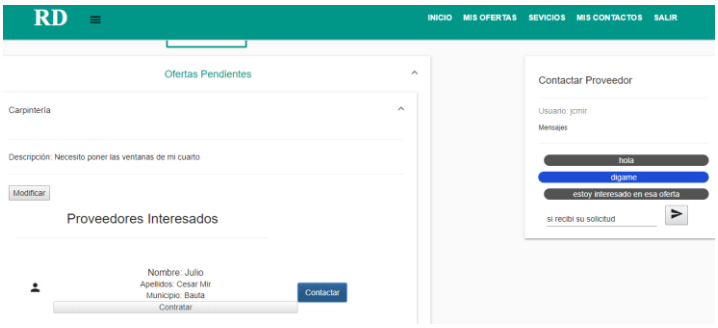
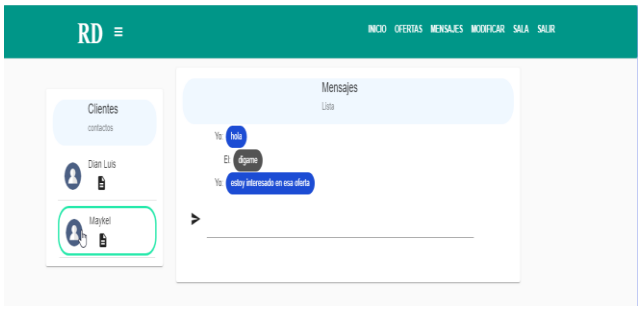
<table>
<tr><td colspan="3">Caso de prueba de integración: Recibir solicitud</td></tr>
<tr><td colspan="3">Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de micro-servicios sobre la plataforma D’Prisa</td></tr>
<tr><td colspan="3">Condiciones de ejecución: Un cliente debe haber creado una oferta.</td></tr>
<tr><td colspan="3">Descripción de la prueba: Comprobar que el módulo es capaz de mostrar los proveedores interesados en la oferta creada.</td></tr>
<tr><td colspan="3">Entradas/Pasos de ejecución:
1. Se selecciona la opción ofertas pendientes.</td></tr>
<tr><td colspan="3">Resultado esperado: El sistema es capaz de mostrar los proveedores interesados en la oferta creada.</td></tr>
<tr><td>Interfaz Cliente:</td><td>Interfaz Proveedor:</td><td>Interfaz Proveedor:</td></tr>
<tr><td> 
</td><td> 
</td><td> 
</td></tr>
<tr><td colspan="3">Evaluación: Prueba satisfactoria.</td></tr>
</table>

Tabla 50: Caso de prueba de integración Paso de mensajes

Caso de prueba de integración: Paso de mensajes	
Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de micro-servicios sobre la plataforma D'Prisa	
Condiciones de ejecución: Un cliente debe haber creado una oferta y un proveedor debe haberla solicitado.	
Descripción de la prueba: Comprobar que el módulo es capaz de enviar, recibir y guardar los mensajes entre un cliente y un proveedor.	
Entradas/Pasos de ejecución: **1. Se selecciona la opción Ofertas Pendientes.** **2. Se escoge a un proveedor interesado.** **3. Se selecciona la opción contactar** **4. Escribir el mensaje deseado y pulsar Enter para enviar.**	
Resultado esperado: El sistema es capaz de enviar y recibir mensajes entre un cliente y un proveedor.	
Interfaz Cliente: 	**Interfaz Proveedor:** 
Evaluación: Prueba satisfactoria.	