UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

“Módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D’Prisa.”

Trabajo de diploma para optar por el título de Ingeniero

en Ciencias Informáticas.

Autor:

Leyan Chang Reyes

Tutores:

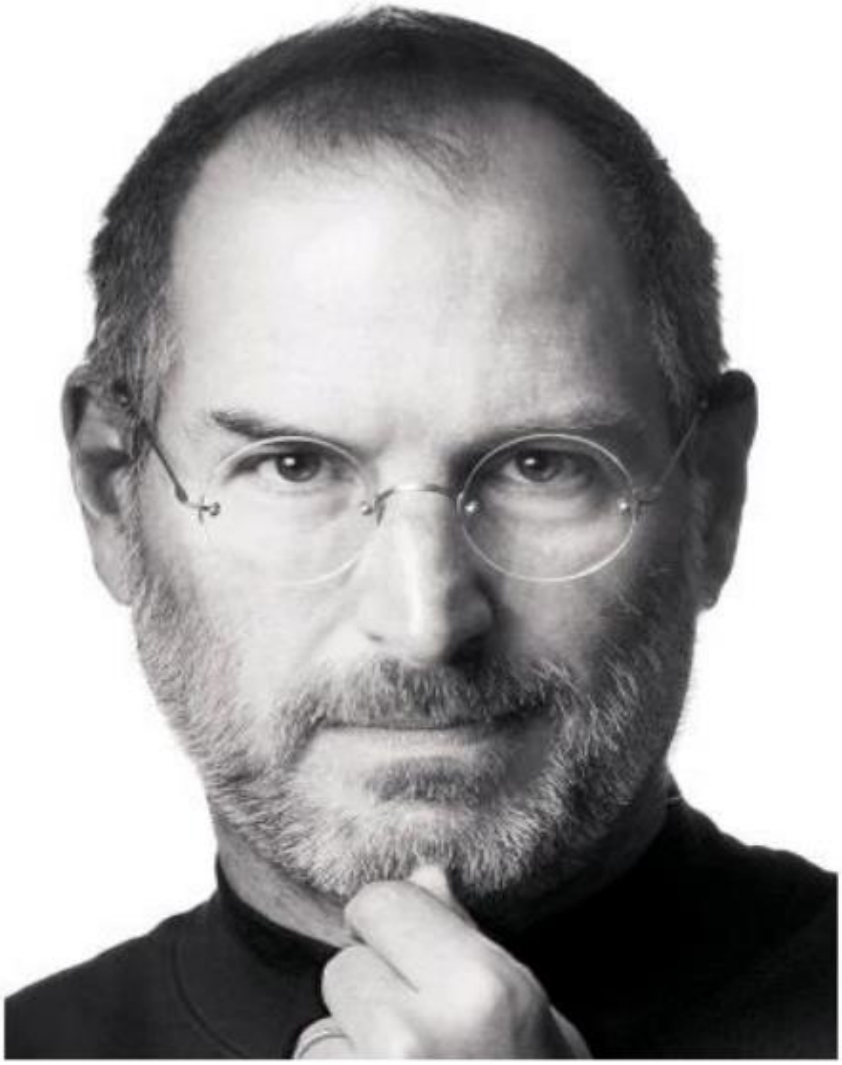
Msc. Joaquín Quintas Santiago

Lic. Lianne Guillén Pérez

La Habana, junio de 2019

¨Año 61 de la Revolución¨

Pensamiento



"Cuando alguien ama lo que hace, se nota. Cuando no amas lo que haces, se nota aún más"

AGRADECIMIENTOS

Quiero agradecer primeramente a mi madre, gracias mima, por tu apoyo en todo momento, por creer en mí y por decirme que todo me saldría bien a pesar de los obstáculos, mami eres la mejor. A mi padre, por ser mi ídolo, mi camino a seguir, por ser y seguir siendo aquel héroe que todo niño desea, pipo te quiero un montón. A mi abuela, por ser la mujer más fuerte que conozco, por haberme educado de la forma correcta y por creer aún que sigo siendo ese pequeño y mimarme tanto, vieja te adoro. A mi hermanito favorito que tanto quiero. A mi familia en general, que tanto apoyo me ha brindado.

A mi novia por su cariño y comprensión.

Gracias le doy a la knoa y mis amigos, por permitirme ser su hermano y compartir tantos momentos.

A mis profesores de la UCI, por ayudarme en mi formación.

A mis tutores por su amistad, guía y apoyo.

En general a todas las personas que, de una u otra forma, colaboraron para que pudiera culminar esta meta tan importante en mi vida.

Declaración de autoría

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

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

Leyan Chang Reyes

_____

Firma del Autor

Lic. Lianne Guillén Pérez

_____

Firma del Tutor

Msc. Joaquín Quintas Santiago

_____

Firma del Tutor

Resumen

Actualmente el sector de las reparaciones domésticas ocupa un lugar en el podio de los sectores más demandados mundialmente. En Cuba ha alcanzado en los últimos tiempos un nivel de demanda elevado. El aumento de esta demanda y los cambios en el modelo económico realizados en el país, han elevado el volumen de personas principalmente en el sector no estatal que se dedican a esta labor. De la mano con este aumento viene asociada la falta de una entidad o sistema que permita escoger al trabajador del sector no estatal que más se ajuste a las necesidades del cliente.

En el presente trabajo se desarrolló un módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre 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, arquitectura de microservicios, proveedores

Resumen

Abstract

The domestic repair sector currently occupies a place on the podium of the most demanded sectors worldwide. In Cuba it has recently reached a high level of demand. The increase of this demand and the changes in the economic model carried out in the country, have increased the volume of people mainly in the non-state sector who are dedicated to this work. Hand in hand with this increase is the lack of an entity or system that allows for the selection of the non-state sector worker that best suits the needs of the client.

In the present work, a module of domestic repair service providers was developed using the architecture of microservices on the D'Prisa platform. It guarantees the interrelationship 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, architecture of microservices, suppliers

Índice

Índice general

Índice de tablas

Índice de figuras

Introducción

Hoy en día se vive en un mundo en el que la continua actualización o cambio del modelo económico de cada país es la clave para su supervivencia económica. En muchos países estas transformaciones han permitido un crecimiento importante del ingreso per cápita, lo que acarrea una mejora en los niveles de vida de los habitantes del mismo, evidenciándose así la importancia de dicho proceso. Con el fin de cumplir este objetivo, Cuba que es una pequeña isla ubicada en el Mar Caribe, se ve envuelta en este proceso de actualización de su modelo económico.

En el desempeño de esta actualización se han producido una gran variedad de cambios. Uno de los más destacados es el de la apertura a reconocer la importancia de otras formas de gestión, paralelas a la gestión estatal, con el objetivo de garantizar la productividad y la eficiencia y en aras de mantener los principios y conquistas de la nación. Esto fue emitido en los Lineamientos para la Política Económica y Social del Partido y la Revolución, aprobados en las sesiones del VI Congreso del Partido Comunista de Cuba, en el año 2011 y renovados en el VII Congreso, en el año 2016.

Con un incremento sostenido en los últimos años, el sector no estatal de la economía o trabajo por cuenta propia ha llegado para quedarse como una eficiente forma de gestión. Cabe resaltar que actualmente este sector goza de gran aceptación en las disímiles esferas y sectores de la sociedad.

El mismo brinda gran variedad de servicios, los cuales varían en diversos factores, destacándose entre estos el factor calidad y el factor precio. Dada esta variedad los clientes de estos servicios, tienen la trabajosa tarea de buscar al cuentapropista que satisfaga sus necesidades y que más se ajuste a sus posibilidades. Prueba indiscutible de esto se encuentra en el sector de las reparaciones domésticas, siendo este una porción que nos toca a todos, debido a que en todos los hogares siempre se requiere o se desea hacer alguna reparación o modificación. Lo que acarrea la difícil tarea de encontrar al trabajador por cuenta propia que más se ajuste a las posibilidades de cada persona. Esta labor se ve obstruida porque la información relativa a los proveedores de reparaciones domésticas no es almacenada y difundida en ningún sitio, trayendo como consecuencias:

Del lado del cliente:

- Dificultad para localizar los proveedores de reparaciones domésticas.
- Dificultad para encontrar la mejor oferta disponible.
- Perdida adicional de tiempo y dinero.
- Dificultad para divulgar la calificación del servicio obtenido de un proveedor determinado.

Del lado del proveedor:

- Dificultad para la divulgación de su trabajo.
- Escasez de ofertas de trabajo disponibles.
- Dificultad para encontrar la mejor oferta de trabajo disponible.

Como era de esperarse estas barreras no solo entorpecen el desarrollo de la tarea de reparación o modificación del hogar, sino que también obstaculizan los servicios brindados por los cuentapropistas.

Dada esta situación y el auge que ha tenido el uso de las TIC en Cuba y en el mundo, la Empresa de Tecnologías de la Información para la Defensa (XETID), dedicada al sector del software, la automática y las comunicaciones, que tiene entre sus principales objetivos la producción de soluciones informáticas, ha visto 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, la cual tiene la peculiaridad de poseer una arquitectura basada en microservicios, los cuales poseen una amplia gama de ventajas, entre las que se destaca su flexibilidad a la hora de crearlos y a la hora de 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 gestión, reseña, clasificación y contratación de servicios.

Para dar solución al problema planteado se define como **objetivo general**: Desarrollar un módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa.

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

Para darle solución al objetivo general se definen los siguientes **objetivos específicos**:

- Establecer el marco teórico-conceptual para fundamentar la investigación a partir del estudio de sistemas reseña, clasificación y contratación de servicios de reparaciones domésticas.
- Realizar el análisis y diseño del módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa, para facilitar la implementación del mismo.
- Implementar el módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa.
- Realizar el proceso de pruebas a la solución propuesta.

Introducción

Para la realización exitosa de la investigación y para darle cumplimiento a los objetivos específicos planteados se ha de llevar a cabo las siguientes **tareas de la investigación**:

1) Análisis de los conceptos fundamentales implicados en el proceso.
2) Realización del estudio del estado de los sistemas de reseña, clasificación y contratación de servicios de reparaciones domésticas.
3) Especificación de las tecnologías, herramientas, metodología y lenguajes necesarios para el desarrollo del módulo.
4) Realización del proceso de identificación, especificación y validación de los requisitos funcionales y no funcionales del sistema.
5) Realización del diseño de la base de datos y otras partes del módulo.
6) Implementación de las funcionalidades del módulo propuesto.
7) Realización de diversos tipos de prueba a la solución propuesta para garantizar su correcto funcionamiento.

Para el desarrollo satisfactorio del presente trabajo de diploma se emplearon los siguientes métodos científicos.

Métodos Teóricos

Método Analítico-Sintético

Se empleó en el momento de buscar la esencia del tema en cuestión, también permitió realizar el análisis de documentos y artículos, realizando una síntesis de los mismos posibilitando así, encontrar los elementos más importantes que se relacionan con el proceso de diseño de sistemas informáticos de esta índole.

Método de Modelación

La utilización de este método posibilitó la modelación de artefactos que describen elementos tales como la base de datos y las clases del módulo.

Métodos Empíricos

Entrevistas

Se utilizó a la hora de realizar entrevistas para obtener del personal calificado, la información necesaria, con el objetivo de enriquecer la solución propuesta.

Investigación documental

Se usó a la hora de realizar el estudio de los documentos que son claves para el desarrollo del sistema.

Introducción

El presente trabajo consta de una introducción, tres capítulos, conclusiones generales, recomendaciones, referencias bibliográficas, glosario de términos y anexos. Los capítulos están estructurados de la siguiente manera:

Capítulo 1. Fundamentación Teórica:

En este capítulo se realiza un estudio sobre los sistemas de reseña, clasificación y contratación de servicios de reparaciones domésticas. Se abordan los conceptos fundamentales a tener en cuenta para comprender los términos usados en el desarrollo del trabajo, se detallan y especifican herramientas, metodología y tecnologías que se utilizarán en la solución final.

Capítulo 2. Propuesta de solución:

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

Capítulo 3. Implementación y pruebas:

En este capítulo se abordan los aspectos relacionados con el desarrollo de la solución propuesta. Se muestran los estándares de codificación aplicados y se describen las pruebas realizadas al módulo, con el objetivo de comprobar el cumplimiento de las tareas de ingeniería planificadas para el desarrollo del mismo.

Capítulo 1: Fundamentación teórica.

El presente capítulo se abordan los temas relacionados con la fundamentación teórica de la investigación. Se detallan los conceptos asociados al dominio del problema, con los cuales se logró una mayor comprensión del objeto de estudio, por parte de los implicados en el desarrollo del módulo. Se realiza un análisis detallado de las tecnologías y herramientas a utilizar. Así como, un estudio del estado del arte del tema de investigación, analizando los sistemas o módulos existentes que son similares al que se va a desarrollar.

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

1.1.1. D'Prisa:

Es una 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 (Santiago, 2018).

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).

1.1.2. MicroServicios:

Un microservicio es un componente desplegable de manera independiente, de alcance acotado y que admite interoperabilidad a través de la comunicación basada en mensajes. (Nadareishvili, y otros, 2015).

A continuación, se exponen varias de las características de un microservicio:

- Son pequeños, independientes y están acoplados de forma flexible.
- Cada microservicio es un código base independiente, que puede administrarse por un equipo de desarrollo pequeño.
- Los microservicios pueden implementarse de manera independiente. Un equipo puede actualizar un servicio existente sin tener que volver a generar e implementar toda la aplicación.
- Cada microservicio es responsable de conservar sus propios datos o estado externo.
- Los microservicios se comunican entre sí mediante API´s.
- No es necesario que los servicios compartan la misma tecnología, bibliotecas o marcos de trabajo.

Los microservicios proponen su propia arquitectura. Mientras que en una arquitectura monolítica la aplicación es desarrollada como una única unidad, una arquitectura de microservicios funciona con un conjunto de pequeños servicios, que se ejecutan de manera independiente y autónoma (chakray.com, 2018).

1.1.3. 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(«Drupal 8 APIs» 2016).

1.2. Análisis de soluciones existentes

En el presente epígrafe se realiza un estudio de sistemas homólogos al que se va a desarrollar. Con el objetivo de identificar las semejanzas y diferencias que pudieran tenerse en cuenta en el desarrollo de la propuesta de solución.

1.2.1. Internacional

Hogar Reparación:

Es una web cuya base de datos dispone de pintores, cerrajeros, técnicos de calderas, mantenimiento de piscinas, jardinería, e instaladores o manitas, se puede acceder a esta web a través de la url: *http://www.hogarreparacion.com/* dentro de su gama de características se destacan (hogarreparacion, 2018):

- Los administradores del sitio se comprometen a atender al cliente en un plazo máximo de 3 horas.
- Brinda gran variedad de servicios, por lo que el cliente puede encontrar la solución a gran variedad de problemas presentes en el hogar.
- Permite listar los servicios por varios criterios como por ejemplo el tipo o la calidad.
- Se tiene la posibilidad de ver una fotografía de la avería que posee el cliente.

Aspectos de interés:

- Su zona de operaciones se restringe a España principalmente en Madrid, Barcelona y Bilbao.
- Su web es intuitiva, sencilla.
- Es un sistema de software privado.
- No opera en Cuba

Sr. Handyman

El Sr. Handyman International, LLC es un negocio de franquicias, con sede en Michigan en los Estados Unidos, que ofrece servicios de mantenimiento y remodelación para propietarios y locales comerciales, se puede acceder a esta web a través de la url: *https://www.mrhandyman.com/ dentro* de su gama de características se destacan (Mr. Handyman, 2018):

- Permite ver la calificación obtenida de los distintos clientes.
- Brinda un servicio de chat en el que los clientes y los proveedores pueden interactuar.
- Brinda gran variedad de servicios, por lo que el cliente puede encontrar la solución a gran variedad de problemas presentes en el hogar.
- Permite listar los servicios por varios criterios como por ejemplo el tipo o la calidad.
- Se tiene la posibilidad de ver una fotografía de la avería que posee el cliente.

Aspectos de interés:

- Su zona de operaciones es principalmente en los Estados Unidos, aunque posee sucursales en otros países como Panamá.
- Su web es intuitiva, sencilla.
- Es un sistema de software privado.
- No opera en Cuba
- 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 alguna labor en el hogar.

UrgeClick

Es una web centrada, principalmente, en las averías urgentes, UrgeClick está formada no solo por profesionales independientes sino también por empresas específicas, talleres y demás, estructurados en hasta 40 categorías distintas, se puede acceder a esta web a través de la url: *http://urgeclick.es/* dentro de su gama de características se destacan (urgeclick, 2018)**:**

- Permite calificar los servicios recibidos.
- Al estar centrado en las averías urgentes los administradores del sitio se comprometen a atender la solicitud en menos de 2 horas.

- Dispone de un servicio de geolocalización para localizar a los profesionales más cercanos.
- Ofrece descuentos y cupones que premian el uso de la plataforma.
- Brinda gran variedad de servicios, por lo que el cliente puede encontrar la solución a gran variedad de problemas presentes en el hogar.
- Permite listar los servicios por varios criterios como por ejemplo el tipo o la calidad.
- Se tiene la posibilidad de ver una fotografía de la avería que posee el cliente.

Aspectos de interés:

- Su zona de operaciones es principalmente en España.
- Su web es intuitiva, sencilla.
- Es un sistema de software privado.
- No opera en Cuba

HomeServe

HomeServe incluye los servicios de albañiles, fontaneros, electricistas, cerrajeros, técnicos del aire acondicionado, la calefacción, pintores, así como carpinteros, cristaleros, instaladores de toldos y más, se puede acceder a esta web a través de la url: *https://www.homeserve.es/* dentro de su gama de características se destacan (homeserve, 2018).

- Brinda gran variedad de servicios, por lo que el cliente puede encontrar la solución a gran variedad de problemas presentes en el hogar.
- Dispone de una sección de seguros de reparaciones.
- Permite listar los servicios por varios criterios como por ejemplo el tipo o la calidad.
- Se tiene la posibilidad de ver una fotografía de la avería que posee el cliente.

Aspectos de interés:

- Su zona de operaciones se restringe a España.
- Su web es intuitiva, sencilla.
- Es un sistema de software privado.
- No opera en Cuba

1.2.2. Nacional

En el ámbito nacional no se encontró ningún sistema para la reseña, calificación y contratación de servicios de reparaciones domésticas.

Con el objetivo de una mejor comprensión del estudio realizado se resume a continuación en la tabla #1, los sitios y las principales características encontradas marcando con una ✔ las que posee y con una ✘ las que no posee.

Tabla 1:Resumen de las aplicaciones existentes

Sistemas/Características		**Hogar Reparación**	**Sr. Handyman**	**UrgeClick**	**HomeServe**
Funcionalidades	Permite ver la calificación recibida.	✘	✔	✔	✘
	Atención Rápida(en menos de 3 horas).	✔	✘	✔	✘
	Listar los servicios por varios criterios.	✔	✔	✔	✔
	Servicio de mensajería.	✘	✔	✘	✘
	Permite ver una imagen de la avería del cliente.	✔	✔	✔	✔
	Brinda gran variedad de servicios.	✔	✔	✔	✔
Aspectos de interés	Software privado	✔	✔	✔	✔
	Sitio intuitivo y sencillo	✔	✔	✔	✔
	Opera en Cuba	✘	✘	✘	✘

Tras el análisis de las soluciones existentes se concluye que ningún sistema es de código de abierto, ni opera en Cuba. Lo que evidencia la necesidad del desarrollo de un nuevo sistema, aunque los mismos brindan las siguientes funcionalidades y aspectos a tener en cuenta para el desarrollo del módulo:

- Establecer un servicio de mensajes con el cual los clientes y los proveedores puedan interactuar.
- Brindar una amplia gama de servicios.
- Brindar una interfaz intuitiva y sencilla.

- Permitir ver una imagen de la avería presentada por el cliente.
- Permitir ver la calificación recibida de los distintos clientes.
- Operar en Cuba.

Con la implementación de estas funcionalidades se prevé, que el módulo posea un alto nivel de usabilidad y un elevado nivel de aceptación por parte de los usuarios.

1.3. Metodología, tecnologías y herramientas utilizadas en la implementación del sistema.

Para la correcta elaboración del módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa, que se ajuste al mundo tecnológicamente cambiante actual, se debe de realizar un profundo estudio sobre las metodologías, las tecnologías y las herramientas existentes, con el fin de escoger las idóneas para el desarrollo. Este estudio se detalla a continuación.

1.3.1. Metodología de desarrollo de software

Una metodología de desarrollo es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. En las dos últimas décadas, respecto a estas metodologías de desarrollo de software se ha entablado un intenso debate entre dos grandes corrientes. Por un lado, las denominadas metodologías tradicionales, centradas en el control del proceso, con un riguroso seguimiento de las actividades involucradas en ellas. Por otro lado, las metodologías ágiles, centradas en el factor humano, en la colaboración y participación del cliente en el proceso de desarrollo y a un incesante incremento de software con iteraciones muy cortas (BusinessSchool, 2016).

Para la selección del enfoque se tiene una serie de indicadores, los cuales son: capacidades y habilidades de desarrollo del equipo de trabajo, el compromiso del cliente, el tiempo de desarrollo, la disposición del equipo ante los cambios durante el proyecto y los recursos materiales disponibles.

En el presente trabajo se utiliza un enfoque ágil, teniendo en cuenta que responde a necesidades y condiciones reales del desarrollo del módulo, entre las que se encuentran:

- La adaptabilidad ante los cambios, ya que el módulo al no tener requisitos bien definidos está a merced de estos.
- El cliente está altamente comprometido y motivado con el desarrollo del sistema.
- El equipo de desarrollo es pequeño y está muy motivado para efectuar el proyecto.
- El tiempo que se dispone para el desarrollo es limitado.

Dentro del catálogo de metodologías agiles existentes se escogió la siguiente:

Programación extrema

La metodología Programación Extrema (por sus siglas en inglés XP) guía el desarrollo de software, haciendo énfasis en las relaciones interpersonales, fomentando el trabajo en equipo y la estrecha comunicación con el cliente. El producto de software se construye teniendo en cuenta que la solución más simple es la mejor. Está orientada a la adaptación paulatina de los requisitos y de sus cambios en cualquier punto de la vida del proyecto. Define historias de usuario para describir las funciones del sistema, las cuales son escritas por el cliente. El ciclo de desarrollo en XP es iterativo e incremental (Calvo, 2018).

Esta metodología aplica un conjunto de prácticas que hacen la entrega del componente menos complicada y más satisfactoria, tanto para los clientes como para el equipo de entrega. Entre las prácticas más importantes se pueden mencionar la codificación por parejas, posibilitando que el código sea discutido y revisado mientras se escribe. Pequeñas entregas del sistema en forma de versiones operativas, aunque no incluyan todas las funcionalidades. Diseño simple y dirección de las pruebas unitarias en la codificación. Además, permite la refactorización del código, dirigida a simplificarlo para facilitar sus cambios en el futuro, presencia continua del cliente en la producción y utilización de estándares de programación, entre otras (Calvo, 2018).

Se escogió esta metodología por los siguientes factores:

- El equipo de desarrollo es pequeño dado que solo consta de un integrante.
- Los equipos XP aceptan los cambios con más facilidad.
- La duración de las iteraciones es corta, lo que posibilita concluir el proyecto en el corto tiempo de desarrollo disponible.
- Los requerimientos que serán desarrollados son priorizados por el cliente.
- XP recomienda las buenas prácticas de desarrollo, como el diseño simple y el desarrollo basado en pruebas.

1.3.2. Lenguaje de modelado

Un lenguaje de modelado es un conjunto estandarizado de símbolos dispuestos para modelar parte de un diseño de software orientado a objetos. Estos se combinan con las metodologías de desarrollo para avanzar de una especificación inicial, a un plan de implementación, para comunicar dicho plan a todo un equipo de desarrolladores. Los modelos proporcionan un mayor nivel de abstracción, permitiendo trabajar con sistemas mayores y más complejos facilitando el proceso de codificación e implementación del sistema de forma distribuida y en distintas plataformas (Fowler y Scott, 1999).

Lenguaje de Modelado Unificado

El lenguaje de modelado unificado (Unified Modeling Language, por sus siglas en inglés UML) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, además aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables(Fowler y Scott, 1999).

Entre sus principales características se encuentran:

- Es independiente del proceso, para utilizarlo de forma óptima se debe usar en un proceso que sea dirigido por Casos de Uso, centrado en la arquitectura e iterativo e incremental.
- Permite modelar sistemas utilizando técnicas de programación orientadas a objetos (POO).
- Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

1.3.3. Herramientas para el modelado

Herramienta CASE

Las herramientas CASE (acrónimo de Computer Aided Software Engineering, en español:(Ingeniería de Software Asistida por Computadora), son diversas aplicaciones informáticas destinadas a aumentar la efectividad en el desarrollo de software y sistemas en términos de productividad y calidad, contribuyendo a una reducción de costos de producción. Incluyen programas para el diseño de sistemas, ingeniería inversa, generadores de código y otros (Beltrán, 20012).

Ejemplos de herramientas CASE se puede mencionar Rational Rose y Visual Paradigm, este último es utilizado para el desarrollo del presente trabajo por lo siguiente.

Visual Paradigm 8.0

Visual Paradigm for UML(VP) es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación y pruebas. Permite construir diagramas

de diversos tipos, código inverso, generar código desde diagramas y generar documentación (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.
- Ingeniería de código: permite la generación de código e ingeniería inversa para los lenguajes: Java, C, C++, PHP, XML, Python, C#, VB .Net, Flash, ActionScript, Delphi y Perl.
- 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.
- Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

1.3.4. 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(ardalis,2018).

Dentro de las tecnologías del lado del cliente se destacan los lenguajes a utilizar los cuales son HTML 5, CSS3 y JavaScript.

HTML 5

HTML5 es el sucesor del HTML4.01 y es el último lenguaje de marcado de hipertexto para los sitios web desarrollados por World Wide Web Consortium (W3C por sus siglas en inglés) y Grupo Web de hipertexto Aplicación de Tecnología de trabajo (WHATWG por sus silgas en inglés). Es el lenguaje de marcado que hace que el proceso de codificación sea más fácil. Permite a los usuarios ejecutar contenidos complejos en plataformas de baja potencia, reducir la necesidad de plugins externos, un mejor manejo de errores. Se utilizará el lenguaje HTML 5 para la creación de las interfaces del sistema, tomando como provecho las peculiaridades del soporte CSS3 y el manejo de formularios en el navegador(Gauchat, 2012).

CSS3

Hoja de estilo en cascada (CSS por sus siglas en inglés), es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas(Gauchat, 2012).

El empleo de hojas de estilo CSS es una práctica que aporta una serie de beneficios como:

- Aumento de la legibilidad y reducción del peso de las páginas Web.
- Mejora del mantenimiento y actualización de los sitios Web.
- Mejora de la accesibilidad: se pueden definir hojas de estilo locales en función de las necesidades o preferencias del usuario.
- Versatilidad: Se ofrecen diferentes hojas de estilo para los diferentes tipos de medio existentes (sintetizadores de voz, dispositivos braille, pantallas de computador a color, televisión).

JavaScript

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web, cuenta con muchas posibilidades, se utiliza para crear pequeños programas. JavaScript es un lenguaje que puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. Con JavaScript se pueden crear diferentes efectos e interactuar con usuarios. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos(Gauchat, 2012).

Ventajas:

- Posee características de interfaz, que son gestionados por el navegador y por el código HTML.
- Los programas JavaScript tienden a ser pequeños y compactos, no requieren mucha memoria ni tiempo adicional de transmisión.
- Es independiente de la plataforma de hardware o sistema operativo, siempre y cuando exista un navegador con soporte JavaScript.

1.3.5. Frameworks del lado del cliente

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(Greene, Caracelli y Graham, 1989).

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

AngularJS

AngularJS v.7.3.3 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. 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).

1.3.6. Tecnologías del lado del servidor

Los lenguajes del lado servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Para ello se utilizan tecnologías que garantizan esta interacción, como es el caso de Node.js(seo, 2018).

Node.js:

Node.js v.8.12.0 es un entorno de ejecución multiplataforma de código abierto para desarrollar aplicaciones web. Esta librería se ejecuta sobre JavaScript y ha sido creada por Google. Es un intérprete Javascript del lado del servidor que cambia la noción de cómo este debería trabajar. Los programadores suelen trabajar con JavaScript del lado del cliente, teniendo que utilizar un lenguaje diferente para el lado del servidor. Ahora, y gracias a Node.js, es posible utilizar el mismo lenguaje de programación para ambos lados, ya que este actúa como un intérprete de JavaScript(Guinard y Trifa, 2016).

Este entorno, que está basado en eventos, utiliza el motor V8 para proporcionar un entorno de ejecución que compila y ejecuta JavaScript a mayor velocidad. Estamos hablando de un entorno de código abierto y multiplataforma, por lo que es posible ejecutar Node.js sin ningún tipo de restricción en Windows, Linux y Mac OS X. Node.js sirve para facilitar la creación de aplicaciones web escalables de manera sencilla y con gran estabilidad, pudiendo ser utilizado para desarrollar cualquier tipo de aplicación(Tilkov y Vinoski, 2010).

1.3.7. Sistema Gestor de Base de Datos

PostgreSQL

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, es 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.
- Permite la gestión de diferentes usuarios, y los permisos asignados a cada uno de ellos.
- Estabilidad.
- Alto rendimiento.
- Flexibilidad: puede funcionar en la mayoría de los sistemas Unix, además de que puede ser integrado a ambientes de Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.

.

1.3.8. Entorno integrado de desarrollo (IDE)

Un entorno integrado de desarrollo (IDE por sus siglas en inglés Integrated Development Environment) es, como su nombre dice, un entorno de desarrollo que incluye un editor para escribir código de programación, que incluye un conjunto de herramientas integradas para la administración de proyectos de software. Algunas de las herramientas incluidas en el IDE ayudan al programador a probar el código en busca de errores (fergarciac, 2013).

NetBeans

NetBeans v.8.1 es un proyecto de código abierto con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de cien socios en todo el mundo. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Tiene soporte para varios lenguajes incluyendo PHP, JavaScript, HTML y CSS (netbeans.org, 2012).

Dentro de las características de NetBeans se pueden encontrar las siguientes:

- Puede conectarse a gestores de bases de datos tales como Oracle, MySQL, PostgreSQL y permite autocompletar los nombres de las tablas, realizar consultas y modificaciones, todo ello integrado en el propio IDE de desarrollo.

- El depurado de las aplicaciones es más sencillo.
- Autocompleta código de los lenguajes que soporta.
- Es multiplataforma

1.3.9. Herramientas para 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. Se usa 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).

Conclusiones parciales

La investigación realizada en este capítulo permitió la exposición de una serie de resultados, que vienen siendo las pautas a tener en cuenta para desarrollar el módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa, dichos resultados serán expuestos a continuación:

- El estudio realizado a las aplicaciones homólogas aportó características importantes a tener en cuenta para desarrollo del módulo.
- Se propone el desarrollo de un módulo sobre la plataforma D´Prisa, aplicando la metodología de desarrollo de software XP, lenguajes como JavaScript, frameworks como Node.js y AngularJS y PostgreSQL como sistema gestor de base de datos.

Capítulo 2: Propuesta de Solución

Introducción

Este capítulo va encaminado a describir la solución propuesta. En el mismo se reflejan las dos primeras etapas de la metodología XP, obteniendo artefactos necesarios para la elaboración del módulo. Por ejemplo, las historias de usuarios, la estimación del esfuerzo y las tarjetas CRC, conformando así la actividad estructural #1 y #2 de la metodología.

2.1. Actividad Estructural #1 Planeación:

La metodología XP plantea la planificación como un dialogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando “Historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”. Una vez obtenidas las “historias de usuarios”, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto, a los efectos de establecer un plan o cronograma de entregas (“Release Plan”), este establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden prestablecido (López, 2016).

2.1.1. Fase de Exploración

Dando comienzo a las fases de la metodología de desarrollo XP está la fase de exploración. En el transcurso de esta fase los clientes plantean a grandes rasgos las historias de usuario, estas constituyen uno de los artefactos más importantes que se genera en la metodología, ya que esta es la forma en que se especifican los requisitos del sistema. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto (López, 2016).

2.1.1.1. Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (López, 2016).

Respecto de la información contenida en la historia de usuario, existen varias plantillas sugeridas, pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción o

sólo una descripción, más quizás una estimación de esfuerzo en días. Beck en su libro presenta un ejemplo de ficha (customer story and task card) en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de seguimiento con la fecha, estado, elementos por terminar y comentarios (López, 2016).

Para esta investigación las historias de usuarios quedan estructuradas de la siguiente manera:

- **Nombre**: Nombre descriptivo de la historia de usuario.
- **Número**: Número consecutivo que identifica a cada historia de usuario.
- **Iteración**: Iteración en la cual será implementada la historia de usuario.
- **Usuario**: El usuario del sistema que utiliza o protagoniza la HU.
- **Prioridad**: Grado de prioridad que le asigna el cliente a la historia de usuario. Los valores que puede tomar son (Alta, Media o Baja).
- **Complejidad**: Grado de complejidad que le asigna el equipo de desarrollo a la historia de usuario luego de analizarla. (Alta, Media o Baja).
- **Descripción**: Descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión.
- **Observaciones**: Condiciones que deben tenerse en cuenta para el desarrollo de la funcionalidad.

A continuación, se describen 3 de las 9 historias de usuario del módulo, el resto pueden ser consultadas en los anexos de la investigación(Ver Anexo 1) .

Tabla 2:Historia de usuario #1 Autenticar usuario

Historia de usuario	
Numero: 2	**Nombre:** Autenticar usuario
Iteración: 1	**Usuario:** Proveedor
Prioridad: Alta	**Complejidad:** Alto
Descripción: El sistema mostrará la interfaz correspondiente, en la cual el usuario tendrá que introducir sus datos (usuario y contraseña).	
Observaciones: Si esta errónea la contraseña o el usuario el sistema mostrará un mensaje de alerta.	

Tabla 3:Historia de usuario #2 Gestionar proveedor

Historia de usuario	
Numero: 1	**Nombre:** Gestionar proveedor
Iteración: 1	**Usuario:** Proveedor
Prioridad: Alta	**Complejidad:** Alta
Descripción: El sistema mostrará la interfaz donde el proveedor podrá introducir sus datos personales y registrarse en el sistema. Ya autenticado el proveedor tendrá la posibilidad de modificar o eliminar su perfil.	
Observaciones: Si los datos de creación o modificación del proveedor no son válidos la aplicación mostrará mensajes con los errores correspondientes. Si todos los datos son correctos el sistema mostrará un mensaje informativo.	

Tabla 4:Historia de usuario #3 Listar oferta

Historia de usuario	
Numero: 3	**Nombre:** Listar ofertas
Iteración: 2	**Usuario:** Proveedor
Prioridad: Media	**Complejidad:** Media
Descripción: El sistema mostrará una interfaz en la cual se listarán las ofertas del sistema.	
Observaciones:	

2.1.2. Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, posteriormente los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días (López, 2016).

La estimación de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la

"velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración (López, 2016).

2.1.2.1. Estimación de esfuerzos

La estimación de esfuerzo se realiza para cada una de las historias de usuario identificadas en la fase de exploración, los resultados se muestran a continuación en la tabla 5.

Tabla 5: Estimación de esfuerzos

Historia de Usuario	Puntos de Estimación
Autenticar usuario	1
Gestionar proveedor	2
Listar ofertas	1
Solicitar oferta	1
Rechazar oferta	1
Ver calificación	1
Filtrar ofertas	1
Establecer servicio de mensajes	1
Establecer sala de chat	1

2.1.3. Fase de Iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios (López, 2016).

Con la descripción de las historias de usuario y la estimación de esfuerzo para cada una se procede a realizar el plan de iteraciones, que son las etapas en las que se dividirá la implementación.

Iteración 1: En esta primera iteración se tiene como objetivo la implementación de las historias de usuario #2(Autenticar usuario) y #1 (Gestionar proveedor), que son las historias de mayor prioridad para el cliente.

Iteración 2: En esta iteración se tiene como objetivo la implementación de las historias de usuario #5(Rechazar oferta), #3(Listar ofertas), #9(Filtrar ofertas) y #4 (Solicitar oferta), las cuales poseen prioridad media.

Iteración 3: En esta iteración se tiene como objetivo la implementación de las historias de usuario #6(Establecer servicio de mensajes), #8(Ver calificación) y #7 (Establecer sala de chat) completando con estas el modulo propuesto.

2.1.3.1. Plan de duración de las iteraciones

Este artefacto tiene como objetivo especificar detalladamente el orden de desarrollo de las HU dentro de cada iteración, así como la duración total de cada una de ellas, brindando así una mayor organización.

Tabla 6:Plan de duración de iteraciones

Iteración	Historia de Usuario	Duración total de Iteraciones
Iteración #1	Gestionar proveedor	3 semanas
	Autenticar usuario	
Iteración #2	Listar ofertas	4 semanas
	Rechazar oferta	
	Solicitar oferta	
	Filtrar ofertas	
Iteración #3	Establecer servicio de mensajes	3 semanas
	Ver calificación	
	Establecer sala de chat	

2.1.3.2. Plan de Entregas

En la tabla # 7 se muestra el plan de entregas, en la que se realiza una propuesta de la fecha aproximada en la que se realizarán iteraciones del módulo.

Tabla 7:Plan de entregas

Iteración	Duración	Fecha de Inicio	Fecha de Fin
Iteración #1	3 semanas	25-2-2019	15-3-2019
Iteración #2	4 semanas	18-3-2019	12-4-2019
Iteración #3	3 semanas	15-4-2019	3-5-2019

2.2. Actividad Estructural #2 Diseño:

La metodología XP hace especial énfasis en los diseños simples y claros, por lo que en esta etapa se define el patrón arquitectónico utilizado en el desarrollo de la solución propuesta, así como la asignación de responsabilidades basada en los Patrones Generales de Asignación de Responsabilidades (GRASP). La metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se emplean otras técnicas como las tarjetas CRC, las cuales serán mostradas posteriormente.

2.2.1. 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 comprender la arquitectura del módulo lo primero que debemos percibir es que, cada microservicio es un sistema independiente, 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 microservicios 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.

En la figura 1 se muestra un ejemplo de la estructura del patrón arquitectónico MVC (Modelo Vista Controlador):

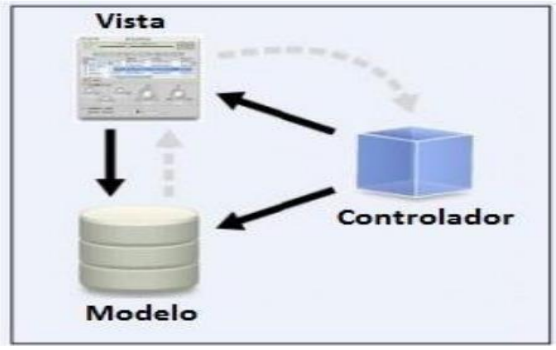


Fig. 1:Patrón arquitectónico MVC(Modelo Vista Controlador)

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 de desarrollo en posteriores proyectos (Recaman, 2012). 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.
- Realizar el de mantenimiento, pues se reduce el tiempo necesario para ello.

Luego de haber conocido la arquitectura de los microservicios a implementar pasaremos a conocer la arquitectura del módulo en general.

Para desarrollo de este módulo se tiene la arquitectura de micro-servicios, conocida 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 microservicios 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, 2016).

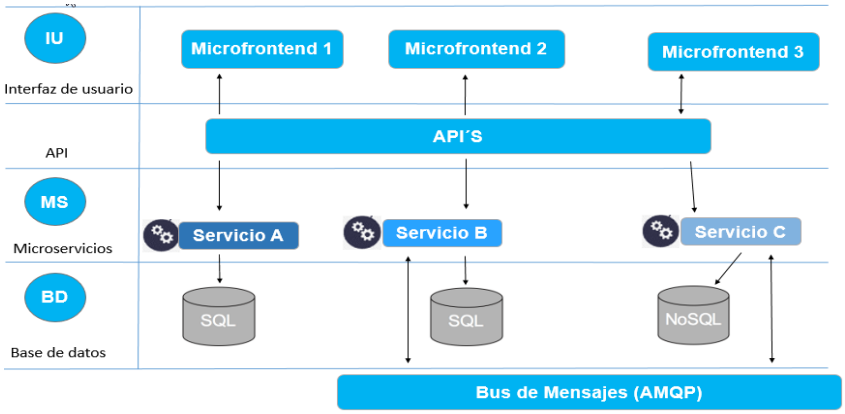


Fig. 2: Arquitectura de microservicios de la plataforma D´Prisa.

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

Ventajas

- **Implementaciones independientes**. Se puede implementar, revertir o poner al día un servicio si algo va mal sin volver a implementar toda la aplicación.
- **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 microservicios 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 microservicios pueden desplegarse según sea necesario, por lo que funcionan bien dentro de metodologías ágiles.

Desventajas

- **Complejidad**. Una aplicación de microservicios 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 microservicio es responsable de la conservación de sus propios datos. Como consecuencia, la coherencia de los datos puede suponer un problema.

2.2.2. Tarjeta Clase Responsabilidad Colaborador (CRC)

La utilización de tarjetas CRC (Class-Responsibility-Collaboration) es una técnica de diseño orientado a objetos propuesta por Kent Beck. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que vamos a necesitar para implementar el sistema y la forma en que van a interactuar, de esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objeto de que el diseño sea lo más simple posible verificando las especificaciones del sistema (López, 2016).

Las tarjetas CRC están esquematizadas de la siguiente forma:

- Nombre de la clase.
- Las responsabilidades de la clase.
- Las clases con las que va a colaborar para poder realizar las responsabilidades indicadas.

A continuación, se describen algunas de las tarjetas CRC del módulo, el resto pueden ser consultadas en los anexos de la investigación (Ver Anexo 2).

Tabla 8:tarjeta CRC #1 Proveedor

Tarjeta CRC	
Clase: Proveedor	
Responsabilidades:	Colaboraciones:
Contener los datos de los proveedores del módulo.	Servicio

Tabla 9:tarjeta CRC #9 ProveedorController

Tarjeta CRC	
Clase: ProveedorController	
Responsabilidades:	Colaboraciones:
registrar (nombre, apellidos, usuario, password, idProvincia, municipio, servicios, provincia): encargado de crear un proveedor. **modificar** (nombre, apellidos, usuario, password, idProvincia, municipio, servicios, provincia, idProveedor): encargado de la modificación de los datos de los proveedores. **eliminar** (idProveedor): encargado de eliminar un proveedor. **getProveedorId** (idProveedor): devuelve un proveedor. **getCalificacion** (idProveedor): devuelve la calificación del proveedor suministrado.	Servicio Provincia Oferta Calificación

2.2.3. 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 los cuales serán descritos a continuación.

2.2.3.1. Patrones GRASP

Los Patrones GRASP (Acrónimo de General Responsibility Assignment Software Patterns) describen los principios fundamentales para asignar responsabilidades a los objetos. A continuación, se describen los patrones de este tipo usados para la realización del módulo.

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 las clases entidad del módulo, ya que cada una constituye un componente especializado en el manejo de datos asociados a la parte del negocio que les corresponde, ejemplo la clase Proveedor y la clase Servicio.

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 dentro del módulo a desarrollar en las clases controladoras, ejemplo ProveedorController y ClienteController.

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 se 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, ejemplo la relación entre la clase Proveedor y la clase Servicio.

2.2.3.2. Patrones GoF

Los patrones GoF pertenecen al campo del Diseño Orientado a Objetos. Están conformados por 23 patrones que se clasifican según su propósito en creacionales, estructurales y comportamiento (Christiansson, 2008).

- Creacionales: definen la forma en la que un objeto puede ser creado teniendo en cuenta la reutilización y mutabilidad. Estos describen la mejor manera de manejar instancias.
- Estructurales: describen cómo los objetos y las clases se pueden combinar para forman estructuras.
- Comportamiento: describen la forma de cómo organizar, administrar y combinar conductas y responsabilidades de objetos, centrándose en la comunicación entre ellos.

Para el desarrollo de la solución se emplearon los siguientes patrones Gof:

Singlenton: El patrón Singlenton hace referencia a aquellos métodos que solo pueden ser llamados si es a través de una instancia de la clase que los contiene (Larman, 2004). Un ejemplo de esto se visualiza en el archivo app.component.ts donde se inyecta el servicio jwt.service.ts, dentro del mismo se encuentran métodos como loggedIn() o logout(), los cuales solo pueden ser llamados si este proceso de inyección es realizado, esto puede apreciarse en la figura 3.

```
logout() {
  localStorage.removeItem('access_token');
  this.router.navigateByUrl('/login');
 }

loggedIn(){
  if (localStorage.getItem('access_token')==null) {
    this.variable=false;
  } else {
    this.variable=true;
  }
  return this.variable;
 }
```

Fig. 3: Métodos de jwt.service.ts

Fachada:

Facade (Fachada): Es la clase definida que ofrece una interfaz común con un conjunto heterogéneo de interfaces. Un ejemplo de ello es la clase Principal que puede apreciarse en la figura 4.

```
40  <div class="main-navigation ">
41    <nav class="navbar navbar-default">
42      <div class="container">
43        <div class="navbar-header">
44          <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
45          <span class="icon-bar"></span>
46          <span class="icon-bar"></span>
47          <span class="icon-bar"></span>
48        </button>
49        <button type="button" mat-button (click)="drawer.toggle ()">
50            <mat-icon>menu</mat-icon>
51        </button>
52        <a class="navbar-brand">RD</a>
53      </div>
54      <div class="collapse navbar-collapse" id="myNavbar">
55        <ul class="nav navbar-nav navbar-right">
56          <li><a [routerLink]="'/body'">Inicio</a></li>
57          <li><a [routerLink]="'/ofertas'">Ofertas</a></li>
58          <li><a [routerLink]="'/mensaje'">Mensajes</a></li>
59          <li><a [routerLink]="'/modificar'">Modificar</a></li>
60          <li><a [routerLink]="'/foro'">Sala</a></li>
61          <li><a (click)="jwt.logout()" (click)="jwt.setControl(0)">Salir</a></li>
62        </ul>
63      </div>
64    </div>
65  </nav>
66 </div>  </div>
```

Fig. 4: Interfaz principal del módulo

2.2.4. Características no funcionales

Las características no funcionales (CNF) son restricciones de los servicios. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. A menudo son aplicados en su totalidad al sistema y normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

La tabla # 12 expone las características no funcionales definidas para el módulo, atendiendo a sus clasificaciones.

Tabla 10: Características no funcionales

Identificador	Descripción
Usabilidad	
CNF# 1	El módulo debe poseer un acceso fácil y rápido.
CNF # 2	Ofrecer interfaz amigable, interactiva e intuitiva.
CNF # 3	El módulo será distribuido en idioma español.
Disponibilidad	
CNF # 4	El servicio web deberá estar disponible las 24 horas del día, asegurando el acceso desde cualquier lugar.
Interfaz	
CNF # 5	La interfaz del módulo contendrá los datos de forma estructurada, permitiendo la interpretación correcta de la información.
CNF # 6	Todos los textos y mensajes en pantalla serán mostrados en idioma español.
CNF # 7	Si ocurre un error con los datos de entrada el módulo muestra mensajes al usuario informando este.
Software	
CNF # 8	Sistema Gestor de Base de Datos PostgreSQL.
CNF # 9	Las PC clientes deben tener instalado un navegador web (Mozilla Firefox, Google Chrome, Opera u otro).
CNF # 10	Arquitectura de microservicios.
Seguridad	
CNF # 11	El módulo debe garantizar la seguridad a través de la autenticación de los usuarios.
CNF # 12	Las diferentes áreas del módulo se encontrarán protegidas contra acceso no autorizado.
CNF # 13	El módulo debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
Hardware	
CNF # 14	Cliente: Procesador Dual Core, 1 GB de RAM, 320 MB de disco duro.
CNF # 15	Servidor: Procesador i3-4170, 4 GB de RAM, 1 TB de disco duro.

Conclusiones Parciales

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

- A partir de la definición de las HU se pudo describir las principales funcionalidades a desarrollar.
- Los artefactos generados según la metodología de desarrollo utilizada y los patrones de arquitectura y diseño descritos, constituyeron una guía fundamental para la construcción de la propuesta de solución.

Capítulo 3: Implementación y pruebas

Introducción

En el presente capítulo se describen los principales detalles de la implementación del módulo. Se especifican las pruebas realizadas al módulo para validar su correcto funcionamiento, con el fin de asegurar que el mismo cumpla con las funcionalidades que precisa el cliente y tenga la calidad requerida.

3.1. Actividad Estructural #3 Codificación:

En esta fase, XP plantea que las HU seleccionadas para ser implementadas se realizan durante el transcurso de la iteración a la que pertenecen. Por estas razones, se lleva a cabo una revisión del plan de iteraciones y se modifican en caso de ser necesario. Como parte de este plan se descomponen las HU en tareas de ingeniería (Joskowicz, 2008).

3.1.1. Estándares de codificación

Las convenciones o estándares de codificación son reglas para escribir el código fuente de manera clara y legible, para facilitar la comprensión y apariencia del mismo. Con el uso de estos estándares se logra elevar la capacidad de mantenimiento del código, sirve como punto de partida para los programadores manteniendo un estilo de programación uniforme y ayuda a mejorar el proceso de codificación haciéndolo en gran medida eficiente y en muchos casos reutilizables (Miranda, 2010).

En función de lograr estandarización en el código del módulo, se definieron las siguientes pautas a seguir durante la implementación:

- **Los métodos se separaron con una línea en blanco.**

```
51
52    funcionMunicipio(prov){
53      this.idprovincia =prov;
54      this.obtProvincias.getMunicipioProvincia(this.idprovincia)
55      .subscribe((result:Municipio[])=>{this.arrayMunicipios = result});}
56
57      addProve(name,apell,user,pass,prov,mun){
58        this.appProveedor.addProveedor(name,apell,user,pass,this.avatar,prov,mun,this.idservicio);}
59
60    ngOnInit() { ···
69    }
70
71    saveData(){console.log(this.datos.value); }
72    step = 0;
73
74    setStep(index: number) {this.step = index; }
75
76    nextStep() {this.step++; }
77
78    prevStep() { this.step--; }
79
80    addService(x){···
92    }
93
```

Fig. 5 : Ejemplo de separación de los métodos

- **Definición de variables**

  Los nombres de las variables deben ser descriptivos y concisos. No se usan abreviaciones. Se utiliza el estilo de codificación “lowerCamelCase”, el cual establece que los nombres inician con letra minúscula y cada nueva palabra debe iniciar con mayúscula.

```
21    isLinear = false;
22    datos: FormGroup;
23    isOptional = false;
24    arrayProvincias: Provincia[];
25    hide = true;
26    arrayMunicipios: Municipio[];
27    idprovincia:number;
28    idservicio:number[]=[];
29    avatar =null;
30    servicio:Servicio;
```

Fig. 6: Ejemplo de variables

3.1.2. Tareas de ingeniería por iteraciones

Las tareas de ingeniería son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, su tiempo de implementación debe ser corto, y su objetivo es resolver las historias de usuario. Una historia de usuario puede tener una o varias tareas de ingeniería, en dependencia de la funcionalidad a desarrollar. Pueden existir también tareas de ingeniería técnicas, que son aquellas que aunque no derivan directamente de una historia de usuario, es necesaria su consideración para que el sistema funcione (Wallace, 2002).

A continuación, se detallan para cada iteración, las tareas a desarrollar por cada HU.

Tabla 11:Tareas de ingeniería por historia de usuario

Nro. de HU	Historias de Usuario	Nro. TI	Tarea de ingeniería por historias de usuario
1	Gestionar proveedor	1	Adicionar proveedor
		2	Modificar proveedor
		3	Eliminar proveedor
2	Autenticar usuario	4	Autenticar usuario
3	Listar ofertas	5	Listar ofertas
4	Solicitar oferta	6	Solicitar oferta
5	Rechazar oferta	7	Rechazar oferta
6	Filtrar ofertas	8	Filtrar ofertas
7	Establecer servicio de mensajes	9	Establecer servicio de mensajes
8	Establecer sala de chat	10	Establecer sala de chat
9	Ver calificación	11	Ver calificación

3.1.2.1. Desarrollo de las tareas de ingeniería

A continuación, se detallan las tareas de ingeniería de la iteración #1, el resto pueden ser consultadas en los anexos de la investigación (Ver Anexo 3).

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

Tabla 12: Tarea de ingeniería # 1

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Adicionar proveedor	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.7
Fecha Inicio: 25 de febrero 2019	**Fecha Fin:** 1 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz donde el proveedor introducirá los datos requeridos logrando con esto adicionarse en el sistema.	

Tabla 13: Tarea de ingeniería # 2

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de usuario**: 1
Nombre de la tarea: Modificar proveedor	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 4 de marzo 2019	**Fecha Fin:** 8 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz donde el proveedor verá sus datos personales y podrá modificarlos a conveniencia.	

Tabla 14: Tarea de ingeniería # 3

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de usuario:** 1
Nombre de la tarea: Eliminar proveedor	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.3
Fecha Inicio: 11 de marzo 2019	**Fecha Fin:** 11 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz donde el proveedor verá su perfil y podrá eliminarse del sistema.	

Tabla 15: Tarea de ingeniería # 4

Tarea de Ingeniería	
Número de tarea: 4	**Número de Historia de usuario:** 2
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 12 de marzo 2019	**Fecha Fin:** 15 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz donde el proveedor introducirá su nombre de usuario y su contraseña y podrá acceder al sistema.	

3.2. Actividad Estructural #4 Pruebas:

Uno de los pilares de XP es el proceso de pruebas ya que anima a probar constantemente o tanto como sea posible (PeteMcBreen, 2002).

XP enfatiza mucho los aspectos relacionados con las pruebas, clasificándolas en diferentes tipos y funcionalidades específicas, indicando quién, cuándo y cómo deben ser implementadas y ejecutadas. Del buen uso de las pruebas depende el éxito de otras prácticas, tales como la propiedad colectiva del código y la refactorización. (Luis Miguel Echeverry Tabón, y otros, 2007)

Según XP se debe ser muy estricto con las pruebas. Sólo se deberá liberar una nueva versión si esta ha pasado el cien por ciento de la totalidad de las pruebas. En caso contrario se empleará el resultado de estas para identificar el error y solucionarlo con mecanismos ya definidos. (Luis Miguel Echeverry Tabón, y otros, 2007)

3.2.1. 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 principales mediante la herramienta Karma y Jasmine, entre estas podemos encontrar la realizada al método getServicio() el cual se muestra en la figura 7:

Método de prueba

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

  const mockResponse =[{"idServicio":1,"tipo":"Albañilería"},{"idServicio":2,"tipo":"Carpintería"},{"idServicio":3,"ti
  const servicioId = [{"idServicio":2,"tipo":"Carpintería"}];

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [ServiciosService]
    });
  });


  describe('Obtener Servicio por id (getServicio (idServicio))', () => {
    it('OK',
    inject([HttpTestingController, ServiciosService], (httpMock: HttpTestingController, myServiceTested: ServiciosServ

      myServiceTested.getServiciosId(13)
      .subscribe(
        (res) => {
          expect(res).toEqual(servicioId);
        }
      );

    })
  );
```

Fig. 7: Código de la prueba unitaria al método getServicio()

3.2.1.1. Análisis de las pruebas unitarias

Se realizaron pruebas al final de cada iteración contando al concluir con las iteraciones con un total de 18 pruebas. Para mayor información sobre la descripción de las iteraciones de prueba ver Anexo 5.

El gráfico que se muestra en la figura 8 muestra los resultados obtenidos en el proceso de pruebas unitarias.

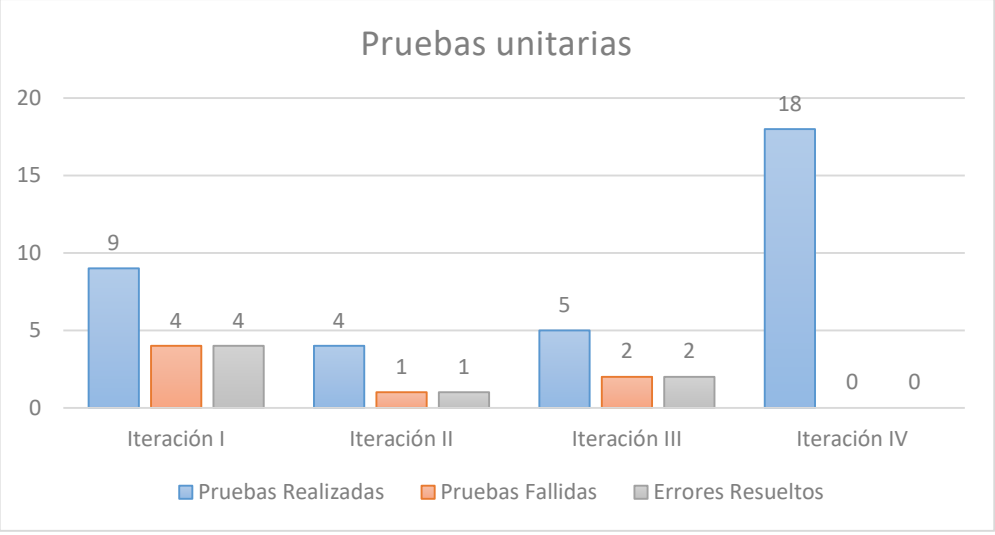


Fig. 8: Resultados de las pruebas unitarias

Como se muestra en la figura 8, se realizaron un total de 18 pruebas unitarias divididas en 4 iteraciones, en la primera de estas iteraciones se realizaron 9 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, el cual fue corregido inmediatamente. En la tercera iteración se realizaron 5 pruebas, donde fallaron 2, las cuales fueron corregidas de inmediato. Tras haber corregido todas las fallas se realizó la última iteración en la que se sondearon todas las pruebas, de estas ninguna falló develando así la correcta implementación de todas las funcionalidades.

3.2.2. 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.

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. Estas marcan 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 (Luis Miguel Echeverry Tabón, y otros, 2007).

Como resultado de las pruebas de aceptación se obtendrán los casos de prueba de aceptación, estos contarán con los siguientes campos:

- **Código:** identificador de la prueba realizada.
- **HU:** número de la historia de usuario a la que hace referencia la prueba a realizar.
- **Nombre:** nombre de la prueba a realizar.
- **Descripción:** se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución:** condiciones que deben cumplirse para llevar a cabo el caso de prueba. Estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba, para que se puedan obtener los resultados esperados.
- **Pasos de Ejecución:** pasos a seguir durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.

- **Resultado esperado:** breve descripción del resultado que se espera obtener con la prueba realizada.
- **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los tres resultados que a continuación se describen:
  - Bien: cuando el resultado de la prueba es exactamente el esperado por el cliente.
  - Regular: cuando el resultado no es completamente el esperado por el cliente de la aplicación y muestra resultados erróneos o fuera de contexto.
  - Mal: cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto.

A continuación, se muestran las pruebas de aceptación realizadas para las historias de usuario de la iteración #1, el resto pueden ser consultadas en los anexos de la investigación (Ver Anexo 4).

Pruebas de aceptación para la Iteración I

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

Tabla 16:Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Gestionar proveedor.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Gestionar proveedor.	
Condiciones de ejecución: En caso de las opciones modificar o eliminar proveedor el mismo debe de existir en la base de datos y estar autenticado.	
Pasos de ejecución: ➢ **Adicionar:** 1. Acceder a la interfaz para registrarse. 2. Completar los campos requeridos con los datos personales. 3. Seleccionar la opción de Registrar. ➢ **Modificar:** 1. Acceder a la interfaz de modificar proveedor. 2. Modificar el o los campos deseados. 3. Seleccionar la opción de Modificar. ➢ **Eliminar:** 1. Acceder a la interfaz de eliminar proveedor. 2. Seleccionar la opción de Eliminar.	

Resultados esperados: Se adiciona, elimina o modifica un proveedor.
Evaluación de la Prueba: Bien.

Tabla 17:Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU2_P2	Historia de usuario: 2
Nombre: Autenticar usuario.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Autenticar usuario.	
Condiciones de ejecución: El proveedor debe de existir en la base de datos.	
Pasos de ejecución: Se llenan los campos usuario y contraseña, luego se selecciona la opción Acceder. (el sistema valida los datos y en caso de ser correctos re-direcciona al proveedor a la interfaz principal.)	
Resultados esperados: Se autentica de forma correcta el proveedor.	
Evaluación de la Prueba: Bien.	

3.2.2.1. Análisis de las pruebas de aceptación

Se desarrollaron un total de 9 de pruebas de aceptación. Estas pruebas fueron realizadas por cada iteración definida. A continuación, se muestra en la figura 9 las pruebas realizadas y las no conformidades en cada iteración.

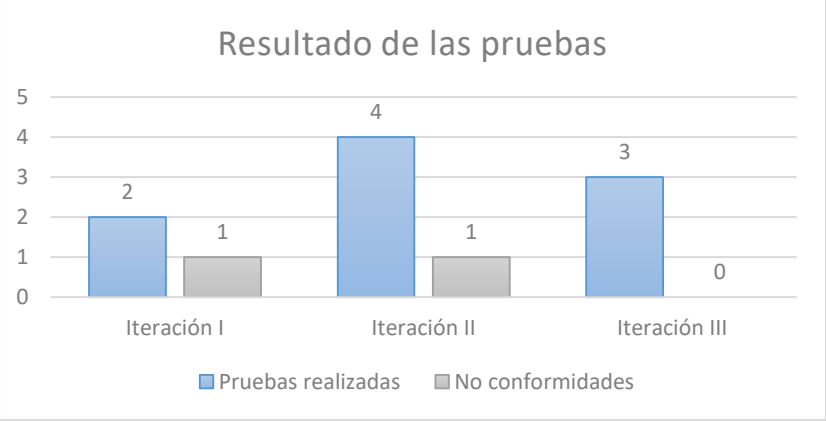


Fig. 9: Resultados de las pruebas de aceptación

Como se muestra en la figura 9, se realizaron un total de 9 pruebas de aceptación divididas en 3 iteraciones, en la primera de estas iteraciones se realizaron 2 pruebas, de ellas 1 falló, obteniendo un 50% de satisfacción. La prueba # 1 detectó que en la opción de modificar proveedor existían campos sin validar, lo

que permitía el paso de datos erróneos a la base de datos, esta no conformidad se corrigió en el menor tiempo posible. En la segunda iteración se realizaron 4 pruebas, de ellas 1 falló, obteniendo un 75% de satisfacción. La prueba # 4 reveló que al solicitar una oferta se eliminaba de la lista de ofertas, pero el cliente no se añadía a la lista de clientes con los cuales puede chatear, esta no conformidad se corrigió inmediatamente después de ser detectada. Por último, en la tercera iteración se realizaron 3 pruebas, las cuales arrojaron resultados satisfactorios. Terminadas estas pruebas y corregidas las no conformidades, el cliente avaló la solución con la entrega del Certificado de Aceptación del Producto, el cual se encuentra en los anexos (Ver Anexo 7)

3.2.3. Pruebas de Integración

Las pruebas de integración son “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 de integración 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:

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

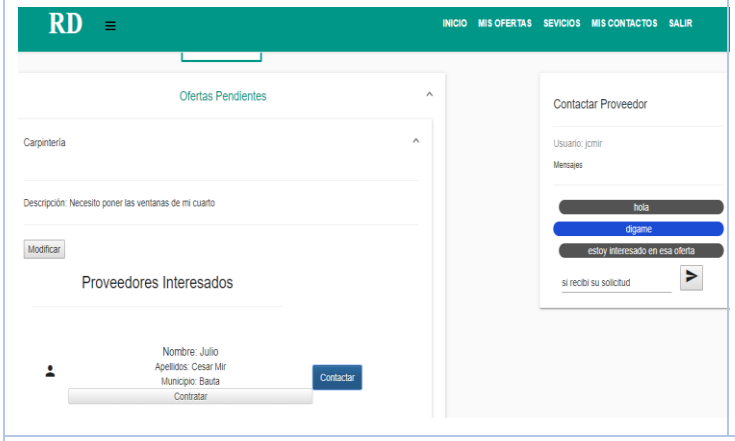
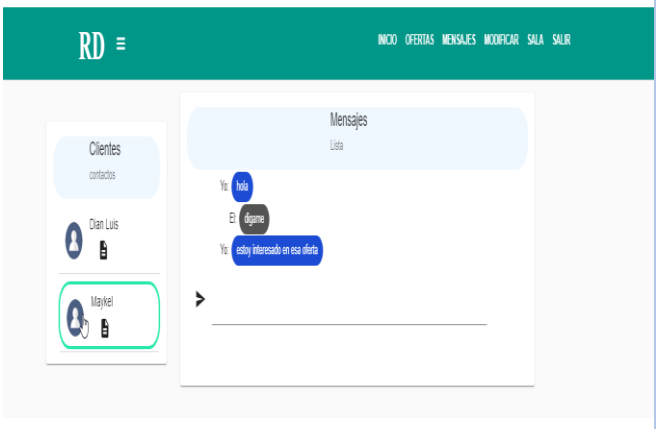
Para la realización de las pruebas de integración del módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D’Prisa, con el sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D’Prisa, se eligió el tipo de prueba incremental, con un enfoque ascendente. Esta 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 reseña, clasificación de servicios y contratación 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, en la tabla 20 se muestra el caso de prueba de integración paso de mensajes. El resto de los casos de pruebas pueden apreciarse en los anexos de la investigación (Ver Anexo 6):

Tabla 18: Caso de prueba de integración Paso de mensajes.

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

Tras haber ejecutado los casos de prueba diseñados para las pruebas de integración, los cuales arrojaron resultados satisfactorios, queda demostrada la correcta integración del módulo elaborado con el sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D'Prisa.

Conclusiones parciales

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

- La ejecución de pruebas permitió detectar las deficiencias presentes, subsanarlas en el menor tiempo posible y ofrecer una aplicación con alto nivel de usabilidad.
- El uso de estándares de codificación permitió generar un código altamente legible.
- El plan de entrega fue cumplido acorde al tiempo planificado para el desarrollo.

Conclusiones generales

Tras el desarrollo del módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios 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 posibilitaron evaluar el producto, comprobándose el correcto funcionamiento del módulo y la ausencia de errores.

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:

- Se recomienda continuar el ciclo de desarrollo del sistema, realizando nuevas iteraciones y agregando nuevas funcionalidades para ganar en usabilidad, logrando así mejorar la competitividad en el mercado del software.

Glosario de términos

Tarjetas C.R.C: Clases Responsabilidades Colaboradores.

UML: Lenguaje Unificado de Modelado o UML, por sus siglas en inglés, Unified Modeling Language.

XP: Programación extrema o eXtreme Programming (XP)

IDE: Entorno de Desarrollo Integrado.

SGBD: Sistema gestor de base de datos.

MVC: Modelo Vista Controlador.

XETID: Empresa de Tecnologías de la Información para la Defensa.

Referencias bibliográficas

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

abc.es. 2015. ¿Qué es una API y para qué sirve? *¿Qué es una API y para qué sirve?* [En línea] abc.es, 15 de 02 de 2015. [Citado el: 13 de 11 de 2018.] https://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html.

aws.amazon. 2018. Amazon API Gateway. *Amazon API Gateway.* [En línea] Amazon Web Services, Inc., 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://aws.amazon.com/es/api-gateway/.

Beltrán, Pedro. 20012. Qué es una herramienta CASE. [En línea] academia.edu, 23 de 3 de 20012. [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.

BusinessSchool. 2016. ¿QuésonlasmetodologíasdedesarrollodesoftwareOBSBusinessSchool? [En línea] 2016. [Citado el: 11 de 1 de 2019.] https://www.obs-edu.com/int/blog-project-management/metodolog%C3%ADa-agile/que-son-las-metodologias-de-desarrollo-de-software.

Calvo, Diego. 2018. Metodología XP Programación Extrema (Metodología ágil). [En línea] 07 de 08 de 2018. [Citado el: 16 de 11 de 2018.] http://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/.

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.

chakray.com. 2018. ¿Qué son los microservicios? *¿Qué son los microservicios?* [En línea] chakray.com, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://www.chakray.com/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/.

Christiansson, Benneth. 2008. *GoF Design Patterns-with examples using Java and UML2.* 2008.

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.

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/.

Fowler, Martin. 2014. Microservices a definition of this new architectural. [En línea] 2014. [Citado el: 15 de 3 de 2019.] https://martinfowler.com/articles/microservices.html.

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

Referencias bibliográficas

hogarreparacion. 2018. Hogar Reparación | Servicio urgente de reparaciones. *Hogar Reparación | Servicio urgente de reparaciones.* [En línea] hogarreparacion.com, 14 de 11 de 2018. [Citado el: 14 de 11 de 2018.] http://www.hogarreparacion.com/.

homeserve. 2018. Reformas integrales, reparaciones y seguros | HomeServe. *Reformas integrales, reparaciones y seguros | HomeServe.* [En línea] homeserve.es, 14 de 11 de 2018. [Citado el: 14 de 11 de 2018.] https://www.homeserve.es:443.

Joskowicz, Ing. José. 2008. *«Reglas y Prácticas en eXtreme Programming». Tesis de mtría.* s.l. : Universidad de Vigo, 2008.

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

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

López, Yolanda Borja. 2015. *Metodología Ágil de Desarrollo de Software.* s.l. : ESPE, MEVAST, 2015.

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

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.

MikeWasson. 2018. Estilo de arquitectura CQRS. *Estilo de arquitectura CQRS.* [En línea] docs.microsoft.com, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/cqrs.

Miranda, Marco. 2010. *Documento de Estándares de Programación.* 2010.

Mr. Handyman. 2018. Mr. Handyman. *Mr. Handyman.* [En línea] mrhandyman.com, 14 de 11 de 2018. [Citado el: 14 de 11 de 2018.] https://www.mrhandyman.com/.

Nadareishvili, Irakli, y otros. 2015. *Microservice Architecture.* 2015. 978-1-491-95625-0.

netbeans.org. 2012. Bienvenido a NetBeans y a www.netbeans.org, Portal del IDE Java Open Source. *Bienvenido a NetBeans y a www.netbeans.org, Portal del IDE Java Open Source.* [En línea] netbeans.org, 2012. [Citado el: 8 de 2 de 2019.] http://netbeans.org/community/articles/welcome-template_es.html.

Newman, Sam. 2015. *Building Microservices.* 2015. 978-1-491-95035-7.

nodejs. 2018. Node.js. [En línea] nodejs.org, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://nodejs.org/es/docs/.

O'Reilly, Sam Newman. 2015. *Building microservices. Designing fine-grained systems.* 2015.

PeteMcBreen. 2002. *QuestioningExtremeProgramming.* s.l. : Addison-WesleyProfessional, 2002.

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.

Referencias bibliográficas

Pressman, Roger S. 2010. *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO.* Nueva York : McGraw-Hill, 2010. 978-607-15-0314-5.

Recaman, Hernando Chaux. 2012. *Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias.* 2012.

Rojas, Carlos. 2018. Micro Frontends. *ion-book.* [En línea] blog.ng-classroom.com, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://blog.ng-classroom.com//blog/tips/micro-frontends/.

Santiago, Joaquín Quintas. 2018. *AyudaDPrisa.* La habana : s.n., 2018.

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

stackshare. 2018. Best Microframeworks (Backend) Software. *Best Microframeworks (Backend) Software.* [En línea] stackshare.io, 13 de 11 de 2018. [Citado el: 13 de 11 de 2018.] https://stackshare.io/microframeworks.

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

urgeclick. 2018. REPARACIONES DEL HOGAR Rápidos y Económicos. *REPARACIONES DEL HOGAR Rápidos y Económicos.* [En línea] urgeclick.es, 14 de 11 de 2018. [Citado el: 14 de 11 de 2018.] http://urgeclick.es/.

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.

ARDALIS, 2018. Tecnologías web comunes del lado cliente. [en línea]. [Consulta: 22 mayo 2019]. Disponible en: https://docs.microsoft.com/es-es/dotnet/standard/modern-web-apps-azure-architecture/common-client-side-web-technologies.

Drupal 8 APIs. *Drupal.org* [en línea], 2016. [Consulta: 12 mayo 2019]. Disponible en: https://www.drupal.org/docs/8/api.

FOWLER, M. y SCOTT, K., 1999. *UML gota a gota*. S.l.: Pearson Educación. ISBN 9789684443648.

GAUCHAT, J.D., 2012. *El gran libro de HTML5, CSS3 y Javascript*. S.l.: Marcombo. ISBN 9788426717825.

GREENE, J.C., CARACELLI, V.J. y GRAHAM, W.F., 1989. Toward a Conceptual Framework for Mixed-Method Evaluation Designs. *Educational Evaluation and Policy Analysis*, vol. 11, no. 3, pp. 255-274. ISSN 0162-3737. DOI 10.3102/01623737011003255.

GUINARD, D. y TRIFA, V., 2016. *Building the Web of Things: With Examples in Node.Js and Raspberry Pi*. 1st. Greenwich, CT, USA: Manning Publications Co. ISBN 9781617292682.

HAHN, E., 2013. *JavaScript Testing with Jasmine*. S.l.: O'Reilly Media, Inc. ISBN 9781449356378.

Referencias bibliográficas

MIKEWASSON, [sin fecha]. Estilo de arquitectura de microservicios - Azure Application Architecture Guide. [en línea]. [Consulta: 12 mayo 2019]. Disponible en: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices.

SEO, 2018. Lenguajes del lado del servidor. *Axarnet* [en línea]. [Consulta: 22 mayo 2019]. Disponible en: https://www.axarnet.es/blog/lenguajes-del-lado-del-servidor/.

TILKOV, S. y VINOSKI, S., 2010. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, vol. 14, no. 6, pp. 80-83. ISSN 1089-7801. DOI 10.1109/MIC.2010.145.

Anexos

Anexo #1 Historias de usuario:

Tabla 19:Historia de usuario #4 Solicitar oferta

Historia de usuario	
Numero: 4	Nombre: Solicitar oferta
Iteración: 2	Usuario: Proveedor
Prioridad: Media	Complejidad: Media
Descripción: El sistema mostrará una interfaz en la cual se podrá solicitar una o más ofertas.	
Observaciones:	

Tabla 20:Historia de usuario # 8 Ver calificación

Historia de usuario	
Numero: 8	Nombre: Ver calificación
Iteración: 3	Usuario: Proveedor
Prioridad: Baja	Complejidad: Baja
Descripción: El sistema mostrará en la interfaz de ver perfil la calificación obtenida de los clientes.	
Observaciones:	

Tabla 21:Historia de usuario #9 Filtrar ofertas

Historia de usuario	
Numero: 9	Nombre: Filtrar ofertas
Iteración: 2	Usuario: Proveedor
Prioridad: Media	Complejidad: Media
Descripción: El sistema mostrará una interfaz en la cual se podrá filtrar las ofertas del sistema por distintos parámetros.	
Observaciones:	

Tabla 22:Historia de usuario #5 Rechazar oferta

Historia de usuario	
Numero: 5	Nombre: Rechazar oferta
Iteración: 2	Usuario: Proveedor
Prioridad: Media	Complejidad: Media
Descripción: El sistema mostrará una interfaz en la cual se podrá rechazar una o más ofertas.	
Observaciones:	

Tabla 23:Historia de usuario #6 Establecer servicio de mensajes.

Historia de usuario	
Numero: 6	Nombre: Establecer servicio de mensajes
Iteración: 3	Usuario: Proveedor
Prioridad: Baja	Complejidad: Alta
Descripción: El proveedor solicitará una oferta de un usuario x y el mismo usuario pasa a formar parte de una lista de clientes con los cuales el podrá comunicarse a través de mensajes.	
Observaciones:	

Tabla 24:Historia de usuario #7 Establecer sala de chat.

Historia de usuario	
Numero: 7	Nombre: Establecer sala de chat
Iteración: 3	Usuario: Proveedor
Prioridad: Baja	Complejidad: Alta
Descripción: Los proveedores tendrán la posibilidad de interactuar con otros proveedores no solo de su localidad si no del país completo.	
Observaciones:	

Anexo #2 Tarjetas Clase Responsabilidad Colaborador:

Tabla 25:tarjeta CRC #2: Cliente

Tarjeta CRC	
Clase: Cliente	
Responsabilidades:	**Colaboraciones:**
Se encarga de contener los datos de los clientes del sistema.	

Tabla 26:tarjeta CRC #10: ClienteController

Tarjeta CRC	
Clase: ClienteController	
Responsabilidades:	Colaboraciones:
getNombre (idCliente): Devuelve el nombre del cliente.	

Tabla 27:tarjeta CRC #3: Servicio

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

Tabla 28:tarjeta CRC #11: ServicioController

Tarjeta CRC	
Clase: ServicioController	
Responsabilidades:	**Colaboraciones:**
getServicio (idServicio): encargada de obtener un servicio por su id. **getServicios ():** encargada de obtener todos los servicios del sistema.	Proveedor

Tabla 29:tarjeta CRC #4: Calificacion

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

Tabla 30:tarjeta CRC #5: Oferta

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

Tabla 31:tarjeta CRC #13: OfertaController

Tarjeta CRC	
Clase: OfertaController	
Responsabilidades:	**Colaboraciones:**
getOfertas (idProveedor): devuelve una lista con las ofertas disponibles para un proveedor. **solicitarOferta ():** solicita la oferta deseada. **rechazarOferta ():** rechaza la oferta deseada. **filtrarOfertas (servico, provincia, municipio):** devuelve una lista filtrada por el o los parámetros suministrados.	Cliente Proveedor

Tabla 32:tarjeta CRC #6: Provincia

Tarjeta CRC	
Clase: Provincia	
Responsabilidades:	**Colaboraciones:**
Se encarga de contener los datos de las provincias.	

Tabla 33:tarjeta CRC #14: ProvinciaController

Tarjeta CRC	
Clase: ProvinciaController	
Responsabilidades:	Colaboraciones:
getProvincia (idProvincia): devuelve el nombre de una provincia. **getProvincias ():** devuelve todas las provincias del sistema.	Proveedor

Tabla 34:tarjeta CRC #7: Municipio

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

Tabla 35:tarjeta CRC #15: ProvinciaController

Tarjeta CRC	
Clase: MunicipioController	
Responsabilidades:	**Colaboraciones:**
getMunicipio (idProvincia): devuelve los municipios de una provincia.	Provincia

Tabla 36:tarjeta CRC #8: Mensaje

Tarjeta CRC	
Clase: Mensaje	
Responsabilidades:	**Colaboraciones:**
Contener los datos de los mensajes del sistema.	

Tabla 37:tarjeta CRC #16: MensajeController

Tarjeta CRC	
Clase: MensajeController	
Responsabilidades:	**Colaboraciones:**
getMnesajes (idProveedor, idCliente, idOferta): devuelve la lista de mensajes entre un proveedor y un cliente. **escribirMensaje (idProveedor, id Cliente, idOferta, mensaje, escritopor):** guarda un mensaje en la base de datos.	Proveedor Cliente Oferta

Anexo #3 Tareas de Ingeniería:

Tabla 38: Tarea de ingeniería # 5

Tarea de Ingeniería	
Número de tarea: 5	Número de Historia de usuario: 3
Nombre de la tarea: Listar ofertas	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 18 de marzo 2019	Fecha Fin: 22 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz con todas las ofertas concernientes al servicio brindado por el proveedor.	

Tabla 39: Tarea de ingeniería # 6

Tarea de Ingeniería	
Número de tarea: 8	Número de Historia de usuario: 9
Nombre de la tarea: Filtrar ofertas	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.8
Fecha Inicio: 8 de abril 2019	Fecha Fin: 12 de abril 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz donde el proveedor podrá escoger uno o varios de los tres criterios (servicio, provincia, municipio) por los que se podrán filtrar las ofertas del sistema.	

Tabla 40: Tarea de ingeniería # 7

Tarea de Ingeniería	
Número de tarea: 6	Número de Historia de usuario: 4
Nombre de la tarea: Solicitar oferta	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 25 de marzo 2019	Fecha Fin: 29 de marzo 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz con las ofertas del sistema las cuales poseen dos opciones, una de ellas es la de solicitar oferta, al presionarlo la oferta desaparece de la lista y el cliente de la oferta pasa a formar parte de la lista de clientes a los cuales el proveedor podrá escribirle algún mensaje.	

Tabla 41: Tarea de ingeniería # 8

Tarea de Ingeniería	
Número de tarea: 7	Número de Historia de usuario: 5
Nombre de la tarea: Rechazar oferta	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 1 de abril 2019	Fecha Fin: 5 de abril 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz con las ofertas del sistema las cuales poseen dos opciones, una de ellas es la de rechazar oferta, al presionarlo la oferta desaparece de la lista de ofertas.	

Tabla 42: Tarea de ingeniería # 9

Tarea de Ingeniería	
Número de tarea: 9	Número de Historia de usuario: 5
Nombre de la tarea: Establecer servicio de mensajes	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 15 de abril 2019	Fecha Fin: 19 de abril 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz con un listado de clientes a los cuales se le podrá enviar uno o más mensajes a los clientes y al mismo tiempo se podrán recibir uno o más mensajes por parte del cliente quedando estos almacenados en la base de datos.	

Tabla 43: Tarea de ingeniería # 10

Tarea de Ingeniería	
Número de tarea: 10	Número de Historia de usuario: 5
Nombre de la tarea: Establecer sala de chat	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 22 de abril 2019	Fecha Fin: 26 de abril 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra una interfaz con una sala de chat en la cual los proveedores podrán ver y enviar mensajes a otros proveedores en tiempo real.	

Tabla 44: Tarea de ingeniería # 11

Tarea de Ingeniería	
Número de tarea: 11	Número de Historia de usuario: 8
Nombre de la tarea: Ver calificación	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 29 de abril 2019	Fecha Fin: 3 de abril 2019
Programador Responsable: Leyan Chang Reyes	
Descripción: Se muestra en la interfaz de ver perfil un apartado en la que se podrá la calificación obtenida de los diferentes clientes.	

Anexo #4 Pruebas de aceptación:

Tabla 45:Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Listar ofertas.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Listar ofertas.	
Condiciones de ejecución: El proveedor debe de estar autenticado.	
Pasos de ejecución: 1. Acceder a la interfaz de ofertas.	
Resultados esperados: Se muestran las ofertas disponibles para el proveedor.	
Evaluación de la Prueba: Bien.	

Tabla 46:Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU9_P8	**Historia de usuario:** 9
Nombre: Filtrar ofertas.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Filtrar ofertas.	
Condiciones de ejecución: El proveedor debe de estar autenticado.	
Pasos de ejecución: 1. Acceder a la interfaz de ofertas. 2. Activar la opción de búsqueda avanzada. 3. Escoger el o los filtros que desee.	
Resultados esperados: Se muestran las ofertas disponibles para el proveedor filtradas por los criterios deseados.	
Evaluación de la Prueba: Bien.	

Tabla 47:Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU4_P4	**Historia de usuario:** 4
Nombre: Solicitar oferta.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Solicitar oferta.	
Condiciones de ejecución: El proveedor debe de estar autenticado y deben existir ofertas disponibles para el proveedor.	
Pasos de ejecución: 1. Acceder a la interfaz de ofertas. 2. Abrir una oferta. 3. Escoger la opción Solicitar.	
Resultados esperados: Se elimina la oferta de la lista y se añade el cliente de la misma a la lista de clientes con los cuales se le puede enviar mensajes.	
Evaluación de la Prueba: Bien.	

Tabla 48:Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU5_P5	**Historia de usuario:** 5
Nombre: Rechazar oferta.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Rechazar oferta**.**	
Condiciones de ejecución: El proveedor debe de estar autenticado y deben existir ofertas disponibles para el proveedor.	
Pasos de ejecución: 1. Acceder a la interfaz de ofertas. 2. Abrir una oferta. 3. Escoger la opción Rechazar.	
Resultados esperados: Se elimina la oferta de la lista.	
Evaluación de la Prueba: Bien.	

Tabla 49:Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU6_P6	**Historia de usuario:** 6
Nombre: Establecer servicio de mensajes.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Establecer servicio de mensajes.	
Condiciones de ejecución: El proveedor debe de estar autenticado y debe haber solicitado previamente una o más ofertas.	
Pasos de ejecución: 1. Acceder a la interfaz de mensajes. 2. Escoger al cliente deseado. 3. Escribirle a ese cliente lo que desee.	
Resultados esperados: Se envía un mensaje al cliente deseado.	
Evaluación de la Prueba: Bien.	

Tabla 50:Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU7_P7	**Historia de usuario:** 7
Nombre: Establecer sala de chat.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Establecer sala de chat.	
Condiciones de ejecución: El proveedor debe de estar autenticado.	
Pasos de ejecución: 1. Acceder a la interfaz de Sala. 2. Escribir lo que desee.	
Resultados esperados: Se envía un mensaje a el resto de los proveedores conectados.	
Evaluación de la Prueba: Bien.	

Tabla 51:Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU8_P9	**Historia de usuario:** 8
Nombre: Ver calificación	
Descripción: Se comprueba el funcionamiento de la historia de usuario Ver calificación.	
Condiciones de ejecución: El proveedor debe de estar autenticado.	
Pasos de ejecución: 1. Acceder a la interfaz de Ver perfil.	
Resultados esperados: Se muestra la calificación obtenida de los distintos proveedores.	
Evaluación de la Prueba: Bien.	

Anexo #5 Pruebas unitarias:

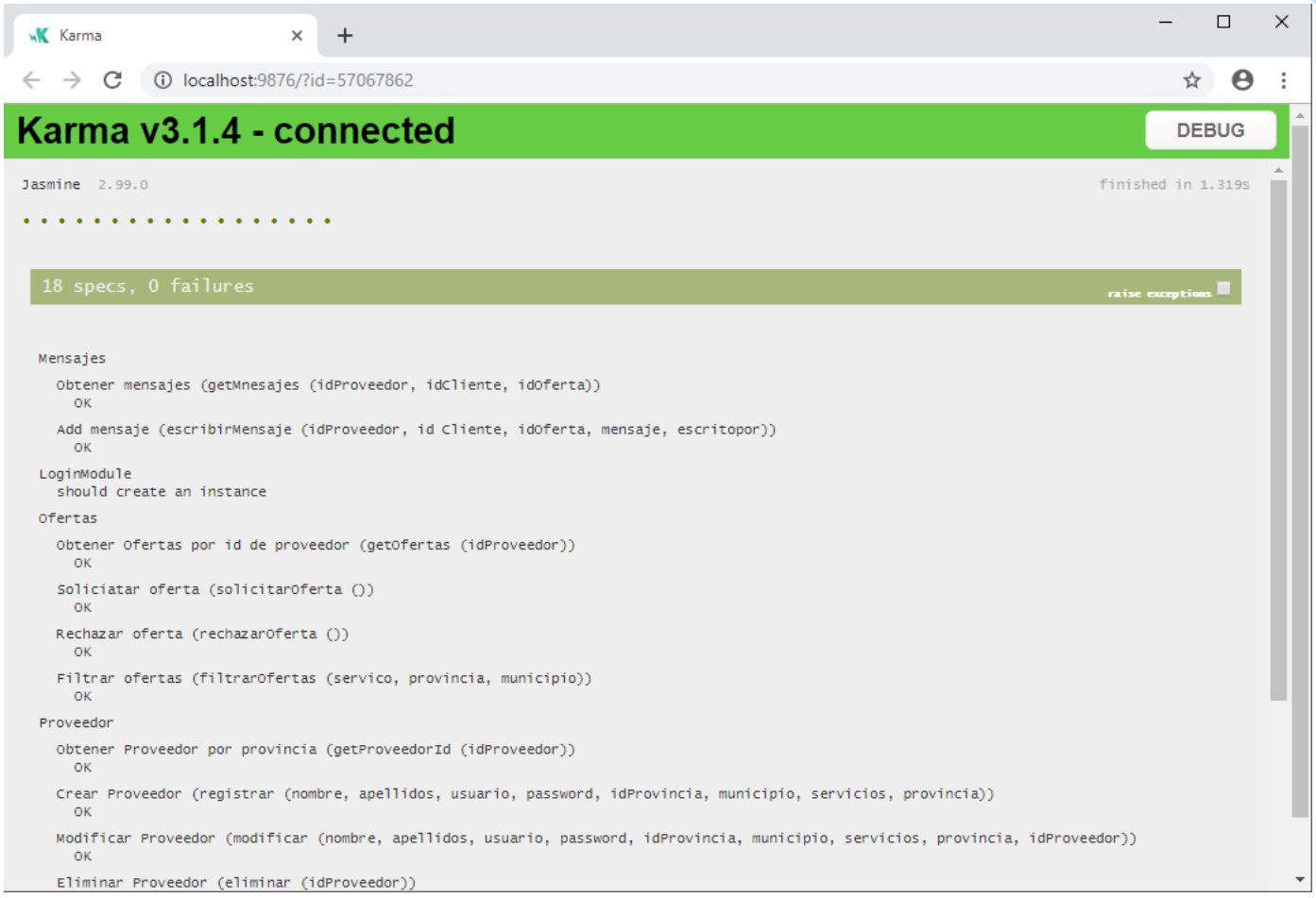


Fig. 10: Resultados de la pruebas unitarias realizadas al módulo.

Anexo #6 Pruebas de integración:

Tabla 52: Caso de prueba de integración Calificar proveedor

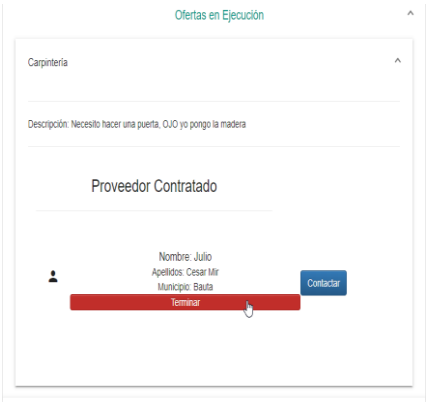
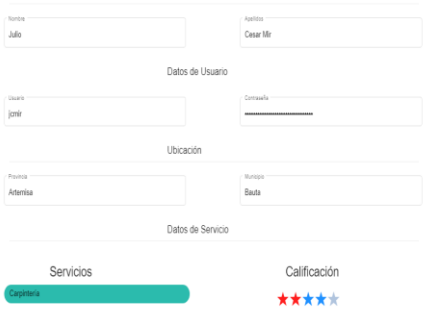
Caso de prueba de integración: Calificar proveedor	
Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D'Prisa	
Condiciones de ejecución: Un cliente debe haber dado por terminada la oferta.	
Descripción de la prueba: Comprobar que el módulo es capaz mostrar la calificación obtenida de los clientes.	
Entradas/Pasos de ejecución: 1. Se selecciona la opción Perfil del menú.	
Resultado esperado: El sistema es capaz mostrar la calificación de un proveedor.	
Interfaz Cliente:	**Interfaz Proveedor:**
 	
Evaluación: Prueba satisfactoria.	

Tabla 53: Caso de prueba de integración Solicitar oferta

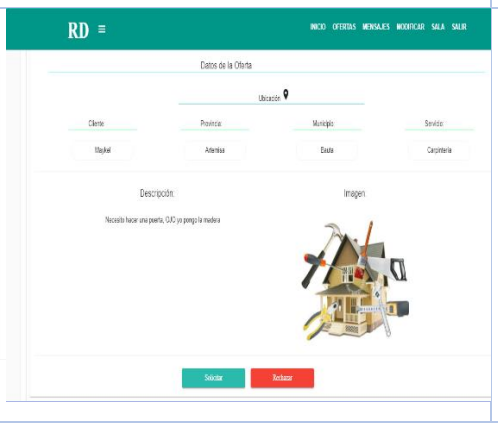
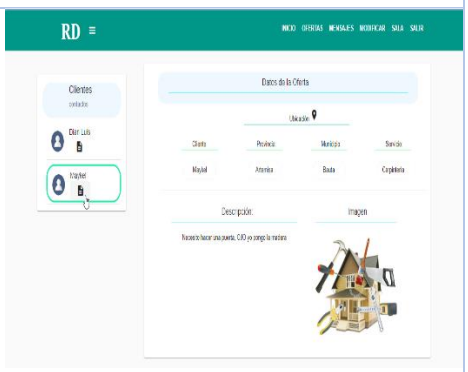
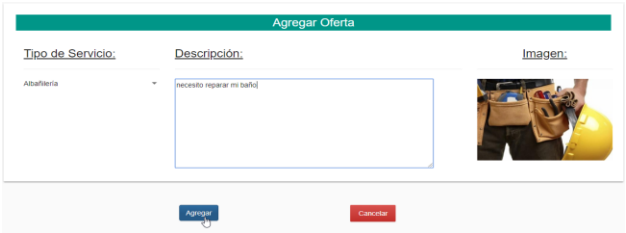
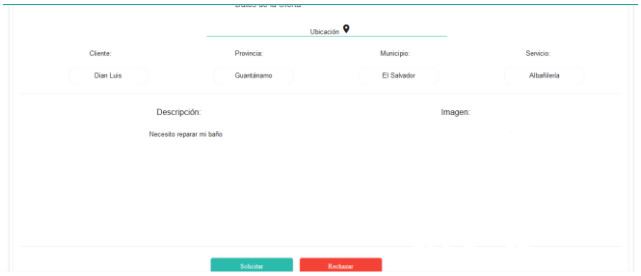
Caso de prueba de integración: Solicitar oferta		
Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D'Prisa		
Condiciones de ejecución: Un cliente debe haber creado una oferta.		
Descripción de la prueba: Comprobar que el módulo es capaz de solicitar una oferta de un cliente.		
Entradas/Pasos de ejecución: 1. Se selecciona la opción ofertas del menú. 2. Se escoge a una oferta. 3. Dar clic en el botón Solicitar.		
Resultado esperado: El sistema es capaz de solicitar una oferta de un cliente.		
Interfaz Cliente:	**Interfaz Proveedor:**	**Interfaz Proveedor:**
		
Evaluación: Prueba satisfactoria.		

Tabla 54: Caso de prueba de integración Recibir oferta

Caso de prueba de integración: Recibir oferta	
Sistema al que se integra: Sistema de reseña, calificación y contratación de servicios de reparaciones domésticas empleando arquitectura de microservicios sobre la plataforma D'Prisa	
Condiciones de ejecución: Un cliente debe haber creado una oferta.	
Descripción de la prueba: Comprobar que el módulo es capaz mostrar las ofertas creadas por los clientes.	
Entradas/Pasos de ejecución: 1. Se selecciona la opción ofertas del menú.	
Resultado esperado: El sistema es capaz de mostrar las ofertas creadas por los clientes.	
Interfaz Cliente:	**Interfaz Proveedor:**
	
Evaluación: Prueba satisfactoria.	

Anexo #7 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 proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa, se hace entrega del producto que se relaciona a continuación:

Módulo de proveedores de servicios de reparaciones domésticas empleando la arquitectura de microservicios sobre la plataforma D'Prisa.

Entrega

Nombre y Apellidos: Leyan Mora Reyes.

Cargo: Estudiante

Firma:

Recibe

Nombre y Apellidos: Ignacio Cuesta

Cargo: D'Prisa Tecnologías

Firma:

Fig. 11: Certificado de Aceptación del Producto