***Asignatura****: TFM-Sistemas Distribuidos*

***Alumno:*** *Agudelo Moncada Carlos Alberto*

***PEC3***

1. ***Introducción***

# *Contexto del Problema*

Con el advenimiento de las Fintech como mecanismo para incluir a todos los ciudadanos en el sistema financiero, abriendo y expandiendo servicios de pago, crédito e inversión, los cuales eran cerradas o destinadas a un cierto sector de la población; nace la oportunidad de crear plataformas que permitan la interacción entre los diferentes participantes de este nuevo ecosistema.

Fintech se define como la industria financiera que aplica las nuevas tecnologías para la prestación de servicios y productos financieros, y de inversión, colocándolos a disposición del público en general vía internet.

De acuerdo con el portal Finnovating.com El crecimiento de las Fintech en Latinoamérica augura un crecimiento vertiginoso. Los expertos estiman que el mercado Fintech superará en Latinoamérica los 150.000 millones de dólares en 2021. Es interesante, ver como en México en los últimos años han nacido más de 300 proyectos de Fintech cifra superada únicamente por Brasil que cuenta prácticamente con 400 startup de este tipo.

Latinoamérica **es un territorio que adapta el pago a plazos con cotidianidad**. No sólo para las grandes compras, también para los pequeños gastos. Esto significa que se genera una cultura del pago aplazado para un gran número de compras, diríamos que casi para todos los ámbitos del consumo. Por tanto, **la adaptación de los sistemas y los nuevos negocios a este modelo de pago es obligatoria**. Las soluciones digitales han revolucionado todo este entorno.

A lo anterior le sumamos que la banca digital reforzó su crecimiento debido a la pandemia que actualmente vivimos. De acuerdo con Asobancaria (Asociación de bancos de Colombia), entre enero y julio de 2020, 1,6 millones de colombianos ingresaran por primera vez al sistema financiero, especialmente, para la recepción de subsidios y ayudas estatales. La transformación digital permitió que Colombia alcanzara las metas de inclusión financiera con 2 años de anciticiapación. Podemos concluir que hoy en día la mayoría de la población en Latinoamérica tiene mecanismos “a la mano” para solicitar y utilizar servicios financieros.

En el sector real, históricamente las PYMES Latinoamericanas y en particular las colombianas conviven con problemas de financiación y flujo de caja. Se ha establecido como política de negociación y pago entre las empresas, que las facturas se paguen 30, 60, 90 y hasta 120 días después de la entrega del producto o prestación del servicio.

Esta práctica o política, que ya es propia del mercado, hace que la empresa tenga que esperar un tiempo importante para recibir el beneficio económico real por su actividad.

Las compañias como alternativa o método para obtener financiación y/o flujo de caja se ven en la necesidad de vender sus facturas por cobrar al sistema financiero quienes, aprovechándose de la necesidad, compran facturas pagando un precio muy inferior o colocando estas facturas como garantes de un crédito. No existe en el mercado una plataforma abierta para la compra y venta de facturas.

Si miramos este escenario desde otro punto de vista, se convierte una oportunidad para que personas naturales y/o jurídicas también puedan obtener beneficio económico en el proceso de compra y venta de facturas, es decir se puede ver que comprar facturas por cobrar se convierte en una inversión a corto y/o mediano plazo.

# *1.2. Propuesta*

De acuerdo con lo descrito en el contexto, mi propuesta como TFM consiste en implementar, sobre una plataforma tecnológica, una solución para la compra y venta de facturas por cobrar, en la cual las empresas puedan ofrecer sus facturas por cobrar y los compradores, a quienes llamaremos inversores, puedan adquirirlas obteniendo una rentabilidad. La solución permitirá adicionar usuarios, inscribir empresas, visualizar los documentos e información de las facturas entre otros.

# *1.3. Objetivos y Requerimientos*

Como objetivos específicos del sistema a desarrollar en este TFM tenemos:

* El sistema deberá permitir el registro de usuarios y/o inversores, permitir el inicio de sesión a los usuarios ya existentes y acceder a su información.
* El sistema deberá permitir la inscripción de empresas y presentar la información de estas.
* La aplicación deberá permitir subir las facturas junto con los documentos de soporte y mostrarla a los inversores interesados.
* La aplicación deberá permitir a los inversores consultar y visualizar información de las facturas y empresas ofertantes.
* La aplicación deberá presentar a los inversores las facturas adquiridas junto con el margen de monetario de utilidad.
* La aplicación presentará la información relacionada con las facturas vendidas por la empresa y el dinero recaudado.

Como objetivo personal busco el consolidar e incrementar los conocimientos adquiridos en las diferentes asignaturas del máster. Específicamente, quiero trabajar temas que son de mi interés profesional y de los cuales no poseo muchos conocimientos como son soluciones de arquitectura con *multiservicios*, *dockers* y *kubernetes*.

El sistema a desarrollar en este TFM debe cumplir las siguientes características:

* Una aplicación móvil o web que permita ejecutar el proceso de compra y venta de facturas por cobrar.
* El sistema se basará en una arquitectura de microservicios. La aplicación tendrá un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros.
* Los diferentes multiservicios se soportarán sobre una infraestructura de contenedores. Utilizaremos Docker para automatizar el despliegue de la aplicación.  Docker nos proporciona una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.
* Exploraremos Kubernetes como orquestador para la automatización del despliegue, ajuste de escala y manejo de aplicaciones en Dockers.

# *1.4. Enfoque y Metodología*

El trabajo por desarrollar consiste en un producto nuevo, que permita a las empresas encontrar un mayor número de compradores para sus facturas por cobrar y así mismo permitir el ingreso de más inversionistas o compradores permitiendo así la apertura a este tipo de producto o negocio.

En este proyecto la metodología a utilizar será de tipo Ágil con el fin de entregar valor al producto en cada uno de los *sprints*. Las **metodologías ágiles mejoran la satisfacción del cliente** dado que se involucrará y comprometerá a lo largo de todo el proyecto. En cada etapa o sprint se informa a los stakeholders, los logros y progresos del proyecto, con la finalidad de involucrarlos directamente para sumar su experiencia y conocimiento, y así, optimizar las características del producto final obteniendo en todo momento una visión completa de su estado. En cada uno de los sprints se detallarán los temas de arquitectura y backlog.

# *1.5. Planificación del trabajo*

El equipo de trabajo está conformado por 2 personas: El estudiante Carlos Agudelo y el tutor Albert Gutiérrez.

Los recursos disponibles para este proyecto son:

* Un laptop Dell con procesador Intel Core i7-7500U CPU 2.70 GHz, sistema operativo Windows 10 y 12 GB de memoria RAM.
* Un laptop Dell con procesador Intel Core i3, sistema operativo Ubuntu 18.04 y 6 GB de memoria RAM
* Una Tablet Huawei con sistema operativo EMUI 8.0 basado en Android 8.0 y con 3 Gb de memoria RAM.

Los detalles de los recursos de software se definirán en el transcurso del proyecto, priorizando el uso de software libre y/o licencia GNU.

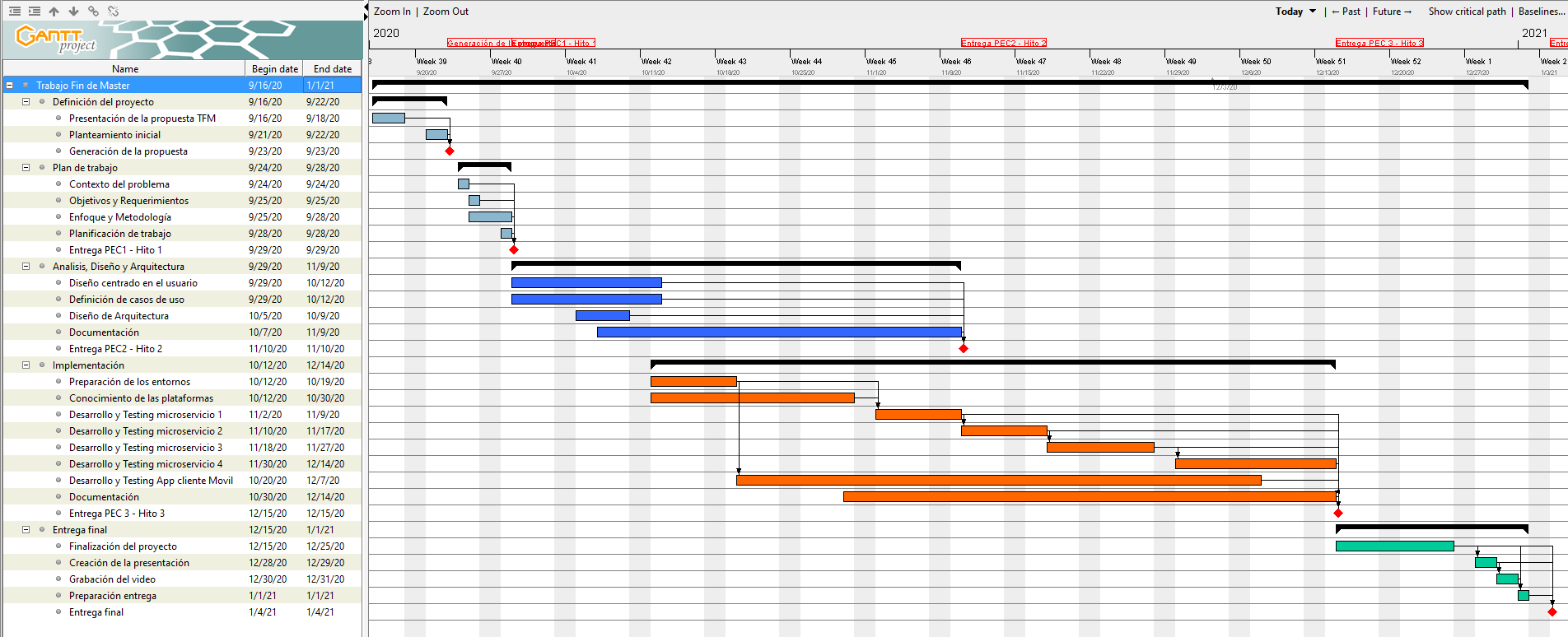
Al utilizar una metodología Ágil, definiremos una planificación de actividades en sprints. El tiempo de cada sprint no deberá ser mayor a 3 semanas. En cada uno de los sprints se realizarán las correspondientes ceremonias con el fin de cumplir con la metodología. La disponibilidad para el proyecto es de 25 horas semanales por lo que tendríamos una planeación temporal así:

|  |  |  |
| --- | --- | --- |
| ***Sprint*** | ***Fecha Inicio*** | ***Fecha Final*** |
| Sprint 0 | 28/09/2020 | 16/10/2020 |
| Sprint 1 | 17/10/2020 | 06/11/2020 |
| Sprint 2 | 7/11/2020 | 27/11/2020 |
| Sprint 3 | 28/11/2020 | 18/12/2020 |
| Sprint 4 | 19/12/2020 | 08/01/2020 |

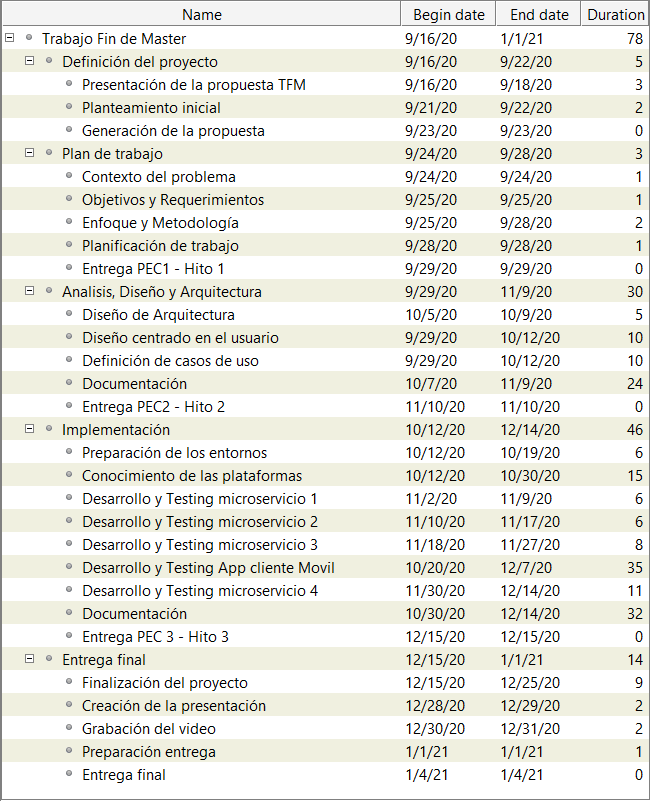
*Tabla 1. Planeación del proyecto por Sprints*

***Nota:*** Es posible que el número de sprints sea mayor dado que algunos pueden ser definidos con menor tiempo dada la complejidad de los ítems del backlog que se quieran implementar. Adicionalmente se define un sprint 0 en el cual se detallarán aspectos como el backlog, arquitectura tecnológica, lenguajes de programación entre otros.

A continuación, se presenta el diagrama de Gantt con las tareas a ejecutar y su estimación:

*Figura 1. Diagrama de Gantt del proyecto*

El listado de las tareas y duración se detalla a continuación:



*Tabla 2. Tareas para ejecutar*

***2. Análisis, diseño y arquitectura***

# *2.1. Diseño centrado en el usuario*

# La experiencia del usuario se refiere a la calidad de la experiencia que tiene una persona cuando interactúa con un diseño específico, originalmente utilizado en las interacciones humano-ordenador. El diseño centrado en el usuario es un proceso de diseño interactivo enfocado en los usuarios, sus necesidades y los objetivos que pretende satisfacer el producto. La usabilidad del diseño es evaluada y mejorada en el proceso.

# Una de las razones por las que se utiliza un Diseño Centrado en el usuario es que este nos brinda los siguientes beneficios:

* Reducir los tiempos de búsqueda y el éxito de las tareas principales.
* Reducir la frustración y conseguir una mayor satisfacción del cliente.
* Disminuir el tiempo de desarrollo de nuestros proyectos.
* Disminuir los costes de mantenimiento y darle mayor vida útil al sitio web.
* Mejorar la credibilidad frente a nuestros diferentes públicos y usuarios.
* Conseguir una mejor alineación entre las metas de negocio y la aplicación.
* Mejorar el valor de marca.
* Aumentar la lealtad y la repetición de la visita.
* Aumentar las páginas vistas y la exposición a contenidos adicionales.
* Disminuir los errores del usuario cuando navega, así como los errores del site o aplicación.
* Mejorar la posición competitiva de una empresa y su posicionamiento en Google.

# *2.1.1. Usuario y Contexto de Uso*

# En esta fase inicial del Diseño Centrado en el Usuario se tiene como objetivo conocer las características de los usuarios, necesidades y objetivos. Realizamos una serie de indagaciones con algunos de ellos quienes nos ayudarán en la obtención de información para la definición del producto y/o servicio.

# Mediante los métodos de indagación, tales como métodos cuantitativos (entrevistas) y cualitativos (encuestas y cuestionarios), obtendremos la data necesaria que nos permita definir las necesidades y preferencias de los usuarios.

# *2.1.1.1. Entrevista en Profundidad*

# La entrevista en profundidad nos permite obtener información de tipo cualitativo. Como lo indicamos en el capítulo 1, es claro que en la solución se presentan 2 tipos de perfil de usuario: el comprador, que a partir de este momento llamaremos *Inversor* y el vendedor quien denominaremos *Ofertante*.

# Realizamos una encuesta inicial a inversores para obtener información valiosa sobre sus comportamientos en la red, hábitos de compra e inversión etc. A continuación se presenta el guion a desarrollar durante la entrevista.

|  |
| --- |
| **Guion** |
| **Información personal** |
| Nombre |
| Edad (Años) |
| Genero |
| Ocupación |
| Ciudad |
| **Hábitos de conexión a internet** |
| ¿Qué experiencia tiene en Internet? |
| ¿Cuántas horas a la semana te conectas a Internet? |
| ¿Desde dónde te conectas habitualmente? ¿Móvil, Laptop? ¿Casa, Trabajo, Calle? |
| ¿Qué sitios visita habitualmente? |
| ¿Cuáles son sus sitios preferidos? |
| **Hábitos de compra e inversión por internet** |
| ¿Has comprado alguna vez en línea? |
| Cuando compras en línea, ¿Cuáles de estas cosas suelen interesarte más? |
| ¿Has realizado algún tipo de inversión de manera virtual? |
| ¿Qué concepto tienes de las operaciones financieras o inversiones en línea? |
| **Aplicación** |
| ¿Haz utilizado alguna aplicación para realizar inversiones? |
| ¿Qué concepto tienen de la aplicación o aplicaciones que utilizas para estas inversiones? |
| ¿Qué tan frecuente utiliza esta aplicación? |
| ¿Conoces alguna aplicación que permita la compra y venta de facturas en línea? |
| ¿Qué funcionalidad deberia incluir? |

# En la primera parte del guion de la entrevista nos enfocamos en obtener los datos básicos del inversor.

# En la segunda parte, le realizamos preguntas para conocer sus hábitos de conexión a internet, como son el número de horas que permanece conectado en la red, desde cual dispositivo, en donde, que sitios visita regularmente, que tipo de información busca o prefiere.

# En la tercera parte indagamos sobre sus hábitos de consumo e inversión en línea y el concepto que tiene de las operaciones financiera o inversiones en línea.

# Por último, preguntamos sí ha realizado inversiones en línea, los conceptos que tiene acerca de las aplicaciones de inversión, la frecuencia con la que las utiliza, si conoce alguna aplicación para la compra y venta de cuentas por cobrar y finalmente, que nos relacione las opciones, que para él debería incluir una aplicación de compra y venta de cuentas por cobrar.

# Debido a la situación actual no todas las entrevistas se han podido realizar de manera presencial. Hemos realizado una de ella a través de teléfono móvil, sin embargo las entrevistas han sido tranquilas con el usuario relajado, sin ningún tipo de presión, sin interrupciones, ni entes externos o terceros que influyan en el desarrollo y/o respuestas dadas por el entrevistado.

# El número de participantes ha sido de 2 voluntarios, seleccionados entre personas conocidas y que tienen ingresos y recursos económicos que indican cierto interés hacia la inversión.

# A continuación relacionamos el resumen de las entrevistas realizadas.

# *Angela:* Mujer de 29 años, soltera, politóloga quien trabaja para entidades gubernamentales. Utiliza el móvil todo la mayor parte del tiempo ya que se ha convertido en una herramienta fundamental en su trabajo. Tiene muy en cuenta las valoraciones que realizan sus amigos y familiares en las diferentes redes *sociales* y aplicaciones.

# Sus hábitos de compra en línea se centraban principalmente en viajes, los cuales realizaba con frecuencia y para lo cual los sitios web de las aerolíneas y otros especializados como despegar, airbnb y tripadvisor se convirtieron en los más visitados. Este año y debido a la pandemia, ha migrado sus hábitos de compra en línea para adquirir productos básicos y de primera necesidad, comida preparada enviada a domicilio y libros impresos. Espera con ansias volver a viajar como lo realizaba normalmente.

# Su principal objetivo de inversión es pagar la hipoteca lo más rápido posible. Conoce varios amigos que realizan inversiones en línea. Ha visitado Forex y tiene conocimiento de cómo funciona pero no ha realizado inversiones por este site. Ha contemplado invertir en bolsa a través de un corredor autorizado y tiene previsto hacer esto en los próximos meses.

# Conoce que las cuentas por cobrar se negocian con los bancos, pero que estos exigen ciertos documentos y requerimientos “muy fuertes” entre otras que la empresa facture muchos millones al mes. Al preguntarle que le gustaría que tuviese la aplicación de este tipo indico:

# Que sea abierta a cualquier inversor.

# Que sea fácil de ingresar y navegar.

# La información a presentar y solicitar sea la absolutamente necesaria.

# Que para las pequeñas empresas también sea asequible ofrecer sus cuentas por cobrar allí porque ella sabe que en tiempos difíciles esto es una opción de financiamiento importante.

# *Diego:* Hombre de 31 años, casado y en espera de su primer hijo, ingeniero de sistemas quien trabaja para una multinacional. Utiliza el móvil todo el tiempo.

# Sus hábitos de compra en línea se centraban principalmente en gadgets tecnológicos, abarrotes, restaurante a domicilio y artículos para recién nacidos.

# Realiza inversiones en finca raíz, invirtió en BitCoin, ha oído hablar del modelo Forex pero no ha realizado inversiones de este tipo.

# Ha escuchado sobre la compra de cuentas por cobrar, ya que un excompañero de colegio que es microempresario ha buscado forma de financiarse mediante este mecanismo pero le ha sido imposible por las barreras y exigencias de los bancos. Indica que como pequeño inversor, este mecanismo puede ser interesante debido a los cortos y medianos plazos que se manejan. Una ventaja es que al ser abierta puede no requerirse mucho dinero para iniciar. Al preguntarle que le gustaría que tuviese la aplicación de este tipo describió:

# Que sea fácil de ingresar y navegar.

# Que sea abierta a cualquier inversor y ofertante.

# Debe presentarse la información referente a fechas, plazos, valores, ganancia de manera clara.

# Que las condiciones de uso y detalles legales no sean tan largos y tediosos.

# Posible contrastar la información con documentos y/o datos de los ofertantes con contactos, teléfonos, email etc.

# Conclusiones:

# Como conclusiones de las entrevistas podemos indicar:

# La utilización del móvil se realiza en cualquier lugar a todo momento y es el principal dispositivo de conexión y consumo de internet.

# Los entrevistados buscan o se documentan sobre diferentes formas de inversión.

# Confían y realizan diferentes transacciones y operaciones financieras en línea.

# Los entrevistados vieron con mucha viabilidad la opción de invertir a través de la solución que estamos construyendo.

# Desean que este tipo de soluciones sea abierta, fácil de usar e interactuar.

# De igual manera que se hizo con el usuario inversor, realizamos una entrevista inicial a empresarios para obtener información valiosa sobre sus comportamientos en la red, ítems de venta y financiación etc. A continuación se presenta el guion a desarrollar durante la entrevista.

|  |
| --- |
| **Guión** |
| **Información personal** |
| Nombre Completo |
| Edad (Años) |
| Genero |
| Ocupación |
| Ciudad |
| **Hábitos de conexión a internet** |
| ¿Qué experiencia tiene en Internet? |
| ¿Cuántas horas a la semana te conectas a Internet? |
| ¿Desde dónde te conectas habitualmente? |
| ¿Qué sitios visita habitualmente? |
| ¿Cuáles son sus sitios preferidos? |
| **Hábitos de venta y financiación por internet** |
| ¿Tú o tu empresa ha vendido o alguna vez en línea? |
| Cuando vendes en línea, ¿Cuáles son los ítems o características en los que más te enfocas? |
| ¿Qué herramientas o a quien buscas cuando necesitas financiación? |
| ¿Alguna vez haz vendido o cedido tus cuentas por cobrar para obtener financiación? |
| **Aplicación** |
| ¿Conoces alguna aplicación que permita la compra y venta de facturas en línea? ¿Cuál (es)? |
| ¿Qué concepto tienes de la aplicación o aplicaciones que permiten este tipo de operaciones? |
| ¿Qué tan frecuente utiliza esta aplicación? |
| ¿Qué funcionalidad debería incluir? ¿Cómo te facilitaría el poder ofrecer y vender tus cuentas por cobrar? |

# En la primera parte del guion de la entrevista nos enfocamos en obtener los datos básicos del ofertante.

# En la segunda parte, le realizamos preguntas para conocer sus hábitos de conexión a internet, número de horas que permanece conectado en la red, desde cual dispositivo se conecta, en donde, que sitios visita regularmente, que tipo de información busca o prefiere.

# En la tercera parte indagamos si venden sus productos o servicios por internet, las características o ítems que prevalecen y aquellos aspectos en los que más hacen énfasis al momento de vender en línea, las herramientas, entidades o mecanismos que buscan cuando requieren financiación y si han acudido a revender sus cuentas por cobrar para obtener recursos económicos con anticipación.

# Por último, preguntamos sí conoce algún tipo de herramienta o solución que le permita ofrecer sus facturas por cobrar, si lo han hecho y que concepto tiene de esas herramientas en caso que las conozca o haya utilizado. Por último queremos conocer su opinión sobre las características que debería tener una solución como esta.

# El número de participantes ha sido de 1 voluntaria, seleccionada entre personas conocidas y que accedió a colaborarnos con el desarrollo de la misma. La entrevistada tiene una pequeño emprendimiento y su perfil encaja en el tipo de usuario al cual la app se enfoca. Esta entrevista ha sido tranquila con la usuaria relajada, sin ningún tipo de presión, sin interrupciones, ni entes externos o terceros que influyan en el desarrollo y/o respuestas dadas por la entrevistada.

# A continuación se describe el resumen de la entrevista realizada.

# *Diana Lucia:* Mujer de 39 años, casada, madre de 2 hijos, ingeniera de sistemas certificada como líder de pruebas, scrum master y product onwer. Ella tiene un pequeño emprendimiento que consiste en asesorar y ofrecer servicios profesionales basada en el conocimiento y experiencia de 15 años como líder de pruebas, analista de requerimientos y product owner a entidades financieras y del sector real. Su emprendimiento ya cuenta con 2 empleados quienes realizan diversas tareas. Utiliza el móvil todo la mayor parte del tiempo ya que se ha convertido en una herramienta fundamental en su trabajo. Tiene muy en cuenta las valoraciones que realizan sus amigos y familiares en las diferentes redes *sociales* y aplicaciones. También está muy al tanto de la evolución de las metodologías, tecnologías y herramientas de la industria.

# Al ofrecer sus servicios, independientemente si el contacto es en línea o presencial, se enfoca en resaltar su conocimiento mediante las certificaciones que ella posee y que son válidas internacionalmente, su experiencia como líder de pruebas, el conocimiento de los sectores y entidades con las que ha trabajado. Como fuente de financiación busca siempre algún tipo de préstamo con entidades financieras y según sus palabras: “uno que otro conocido”.

# Conoce que las cuentas por cobrar se negocian con los bancos y por su conocimiento de los productos financieros, las condiciones y requisitos para ella son imposibles. Al preguntarle que le gustaría que tuviese la aplicación de este tipo indico:

# Que sea totalmente abierta a cualquier pyme o emprendimiento.

# Que las condiciones para ingresar no sean una barrera imposible de superar.

# Que tenga una buena usabilidad.

# La información a presentar y solicitar sea la absolutamente necesaria.

# *2.1.1.2. Encuesta*

# Adicional a las entrevistas en profundidad, se realizaron encuestas con el fin de obtener información de tipo cuantitativo. Algunas preguntas dependen de la respuesta de otras, sin embargo el objetivo es conocer en un rango un poco más amplio data valiosa para la definición y creación de nuestra solución. Al igual que con la entrevista, se crearon 2 formularios o tipos de encuesta. Uno para el inversor y otro para el ofertante.

# A continuación, presentamos el formulario de encuesta para el inversor.

|  |  |
| --- | --- |
|  |  |

# Como se observa en la imagen anterior, con las primeras preguntas se busca obtener información socio-demográfica como son el rango de edad, ciudad de residencia, uso del móvil etc. Posteriormente queremos conocer si le atrae realizar inversiones en productos financieros y si buscan información para realizar inversiones.

# 

# En las últimas preguntas se indaga por el tipo de inversiones que busca, si ha realizado inversiones de manera virtual y el concepto que tiene de las operaciones o inversiones en línea.

# De igual forma se desarrolló un formulario de encuesta para el perfile Ofertante. Como se observa en la imagen a continuación, con las primeras preguntas se busca obtener información socio-demográfica como son el rango de edad, ciudad de residencia, uso del móvil etc. Posteriormente queremos conocer los ítems o características en los que más se enfoca al momento de vender en línea y si utiliza el móvil para obtener información o posibilidades de financiación.

|  |  |
| --- | --- |
|  |  |

# En las últimas preguntas se indaga si ha vendido o cedido las cuentas por cobrar para obtener financiación y si conoce alguna aplicación que permita hacer este tipo de negociaciones u operaciones.

# 

# *2.1.2. Perfiles de usuarios*

# Como se ha descrito en los numerales anteriores, se han identificado 2 tipos o grupos de usuarios para la aplicación: Estos perfiles están basados en las características, aspectos socio-demográficos, necesidades y expectativas de uso. Al ser una plataforma para compra y venta de cuentas por cobrar se definen 2 tipos que el inversor y el ofertante. El detalle identificado de estos 2 perfiles de usuario, durante la fase de análisis utilizando las técnicas de indagación son:

|  |  |
| --- | --- |
| *Usuario Inversor* | |
| *Características del perfil* | A este perfil corresponden las personas que desean hacer una inversión para obtener algún tipo de rendimiento o ganancia a mediano y corto plazo, comprando cuentas por cobrar a ofertantes que las publican. La diferencia entre el valor de la cuenta o factura y el valor de lo que el inversor paga corresponde a la ganancia o rentabilidad obtenida.Estos usuarios son mayores de edad que tienen un dinero para invertir y desean obtener rentabilidad rápidamente. No hay un rango de edad definido pero se estima que la mayoría de los inversores se encuentre entre los 25 y 45 años de edad, debido al uso que tienen del móvil y la confianza que tienen sobre las transacciones que se realizan en internet.Estos usuarios ingresaran a la aplicación en busca de oportunidades de inversión que cumpla con sus expectativas como inversor. |
| *Contextos de Uso* | Los usuarios inversionistas utilizarán la aplicación desde cualquier lugar donde se encuentren: casa, oficina, transporte público, un café etc. La utilizarán desde el móvil y a cualquier hora del dia. Ingresarán cuando cuenten con recursos para realizar una inversión, para visualizar las facturas compradas con el fin de cobrar las que se vayan venciendo y actualizar su perfil de usuario. |
| *Tareas* | Registrarse en la appIniciar sesiónBuscar oportunidades de inversiónVer en detalle cada una de las oportunidadesComprar una cuenta por cobrarVisualizar el detalle de las cuentas compradasActualizar sus datos |

|  |  |
| --- | --- |
| *Usuario Ofertante* | |
| *Características del perfil* | A este perfil corresponden las entidades y personas que ofrecen y/o publican cuentas por cobrar para que los inversionistas las adquieran.Estos usuarios corresponden a empresas que necesitan financiación o recursos económicos rápidamente y toman la decisión de vender sus cuentas por cobrar, por un valor menor al establecido en la factura. No hay un determinado sector, tipo de empresa o industria a la cual este orientado la aplicación. Sin embargo se estima que la mayoría de los ofertantes corresponden a Pymes.Estos usuarios ingresaran a la aplicación y ofertaran sus cuentas por cobrar en busca de inversores y flujo de caja. |
| *Contextos de Uso* | Los usuarios ofertantes utilizarán la aplicación desde cualquier lugar donde se encuentren: casa, oficina, transporte público, un café etc. La utilizarán desde el móvil y a cualquier hora del dia. Ingresarán cuando necesiten o requieran ofrecer sus cuentas por cobrar, para visualizar las facturas vendidas y actualizar su perfil de usuario. |
| *Tareas* | Registrarse en la appIniciar sesiónOfrecer o publicar sus cuentas por cobrarVer en detalle cada una de las oportunidades publicadasVisualizar el detalle de las cuentas vendidasActualizar sus datos |

# *2.1.3. Historias de Usuario*

# Una historia de usuario es una representación de un [requisito](https://es.wikipedia.org/wiki/Requerimiento_(sistemas)) escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las [metodologías de desarrollo ágiles](https://es.wikipedia.org/wiki/Metodolog%C3%ADa_%C3%A1gil) para la especificación de requisitos.

# Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes. Las historias de usuario que hemos definido son:

***Login***

***Escenario 1: Login de usuario***

**Dado** un usuario que posse la app   
**Cuando** abre la aplicación  
**Entonces** le debe mostrar la pantalla de login que tiene los siguientes componentes

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | “Log in” |
| Correo | * Max 120 caracteres. * El campo debe ser editable. * Campo requerido |
| Contraseña | * Permitir máximo 12 caracteres. * Permitir números y caracteres especiales. * Debe mostrarse enmascarado. |
| Log in | * Botón (para esta historia no tendrá funcionalidad) |

***Escenario 2: Usuario No se encuentra registrado en el app***

**Dado** un usuario que posee la app   
**Y** el usuario No ha realizado el registro previamente  
**Y** digita el usuario y la contraseña   
**Cuando** le da Tap en Log in  
**Entonces** debe mostrar la pantalla de registro que tiene las siguientes características

|  |  |  |
| --- | --- | --- |
| ***Campo*** | ***Tipo*** | ***Características*** |
| Nombres | Texto | * Max 120 caracteres, No permite números y caracteres especiales. * Campo requerido. |
| Apellidos | Texto | * Max 120 caracteres, No permite números y caracteres especiales. * Campo requerido. |
| Email | Texto | * Debe mostrar precargado el email ingresado por el usuario en la página de login. * Max 120 caracteres. * El campo debe ser editable. * Campo requerido. |
| Contraseña | Texto | * Permitir máximo 12 caracteres. * Permitir números y caracteres especiales. * Debe mostrarse enmascarado. |
| Confirme contraseña | Texto | * Permitir máximo 12 caracteres. * Permitir números y caracteres especiales. * Debe mostrarse enmascarado. * Se requiere validar que coincida con el password ingresado por el usuario en el campo “Contraseña”. |
| Desea registrarse como | Checkbox con dos opciones Inversionista, ofertante. | Solo debe poderse marcar una opción. |
| *Botón* | *Siguiente* | *Cuando el usuario le da TAP debe enviarlo a la pantalla correspondiente para finalizar el registro dependiendo de la opción seleccionada en el campo anterior “Debe registrarse como”.* |

***Escenario 3: Usuario registrado previamente***

**Dado** un usuario que posee la app   
**Y** ya ha realizado el registro completo previamente  
**Y** digita el usuario y la contraseña  
**Cuando** le da Tap en el botón Log in  
**Entonces** le debe mostrar la pantalla de home correspondiente (Inversor/ Ofertante)

***Escenario 4: Usuario registrado con email y contraseña, pero no ha finalizado su registro como Inversor u Ofertante***

**Dado** un usuario que posee la app   
**Y** ya ha realizado el registro del correo y la contraseña  
**Y** no ha finalizado su registro como Inversor u ofertante  
**Y** digita el usuario y la contraseña.  
**Cuando** le da Tap en el botón Log in  
**Entonces** le debe mostrar la pantalla de registro correspondiente para Inversor/ Ofertante

***Registro Inversor***

***Escenario 1: Pantalla registró inversor***

**Dado** un usuario que posee el app  
**Y** está en la pantalla de registro de usuario y contraseña  
**Y** se ha registrado exitosamente el usuario y la contraseña  
**Y** selecciono Inversor como opción del registro  
***Cuando*** de tap en el botón siguiente  
***Entonces*** debe llevarlo a la pantalla de Registro de Inversor que tiene los siguientes componentes:

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | “Registro Inversor” |
| Nombre y Apellidos | Precargados No editable |
| Tipo de identificación | Dropdown list con los tipos de identificación que apliquen para persona natural.   * Campo requerido |
| Número de identificación | Debe permitir máximo 20 caracteres.   * Campo requerido |
| Monto inicial de la inversión | No debe permitir ingresar letras.   * Campo requerido |
| Moneda | Dropdown list, con los valores   * COP * EUR * USD   Solo se puede seleccionar uno   * Campo requerido |
| Número de celular | Debe permitir máximo 15 caracteres. Solo debe permitir ingresar Números.   * Campo requerido |
| Aceptar términos y condiciones | Checkbox   * Campo requerido |
| Finalizar registro | Botón (para esta historia no tendrá funcionalidad) |

**Pantalla de Home Inversor**

***Escenario 1: Pantalla de Menú inversor***

**Dado** un usuario que posee el app  
**Y** se ha finalizado el registro como inversor de manera exitosa  
**Y** hace login en la aplicación   
**Cuando** hace tap en el botón login  
**Entonces** le debe mostrar la pantalla de Home del inversionista con los siguientes campos:

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Menú inversor |
| Texto | “Bienvenido al home del inversionista” |
| Icono + Texto | Icono de búsqueda de oportunidades + “Oportunidades”. Sin funcionalidad para esta historia |
| Icono + Texto | Icono de Mis inversiones + “Mis inversiones”. Sin funcionalidad para esta historia |
| Icono + Texto | Icono de cuenta + “Mi Cuenta”. Sin funcionalidad para esta historia. |

**Búsqueda de Ofertas**

***Escenario 1: Pantalla de búsqueda de oportunidades***

**Dado** que posee el app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Cuando** selecciona la opción de ofertas  
**Entonces** le debe abrir la pantalla para búsqueda de ofertas, con la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Buscar |
| Texto | “Busca aquí las mejores oportunidades, ingresa el monto mínimo que estás dispuesto a invertir. Te mostraremos las ofertas que sean iguales o mayores al monto ingresado.” |
| Textbox Monto | El usuario podrá digitar el valor, solo se permite el ingreso de números. Y es un campo requerido |
| Moneda | Dropdown list con las opciones de monedas   * COP * EUR * USD   Solo podrá seleccionar uno   * Campo requerido |
| Botón | “Buscar” Cuando el cliente le de tap deberá ir a la pantalla para listar las ofertas de acuerdo al filtro. |

**Listado de Ofertas**

***Escenario 1: lista de ofertas***

**Dado** que posee el app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Y** selecciona la opción de ofertas  
**Y** se muestra la pantalla de búsqueda de ofertas  
**Y** digita la información requerida  
**Cuando** da TAP en el botón buscar  
**Entonces** le debe mostrar la lista de ofertas que cumplan con los criterios de búsqueda, mostrando la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Oportunidades |
| Icono | Icono definido |
| Información de la oportunidad | * Valor * Plazo * Fecha de cobro |
| Botón | “Consultar” Cuando el cliente le de tap deberá ir a la pantalla del detalle de la factura. |
| Botón | Back  Cuando el cliente de tap en back debe volver a la página de búsqueda. |

***Escenario 2: No hay resultados para la búsqueda***

# Dado que posee el app Y se ha iniciado sesión correctamente como Inversionista Y va al Home del inversionista Y selecciona la opción de ofertas Y se muestra la pantalla de búsqueda de ofertas Y digita la información requerida Y no hay resultados para la búsqueda Cuando da TAP en el botón buscar Entonces le debe mostrar un mensaje que le indique que no hay resultados para la búsqueda “Lo sentimos, no encontramos ofertas para el monto y moneda ingresados”

**Pantalla detalle de la oportunidad**

***Escenario 1: Pantalla detalle de la oportunidad/oferta***

**Dado** que posee la app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Y** selecciona la opción de ofertas  
**Y** se muestra la pantalla de búsqueda de ofertas  
**Y** digita la información requerida  
**Y**  hay resultados para la búsqueda  
**Cuando** da TAP en el botón Consultar  
**Entonces** le debe mostrar la pantalla de detalle de la oferta con la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Detalle |
| Nombre del ofertante | * Nombre del ofertante precargado. * NO editable. |
| Valor de la factura | * Valor de la factura. * NO editable. |
| Moneda de la factura | * Moneda de la factura. * NO editable. |
| Fecha de cobro | * Formato dd/mm/aaaa * No editable. |
| Días restantes para cobro | * Días. * No editable. |
| Detalle/observaciones | * Texto si aplica. * No editable. |
| Botón | * Regresar. * Cuando el cliente da TAP lo lleva a la página anterior. |
| Botón | * Comprar * Cuando el cliente da TAP lo lleva a la página. |

**Pantalla de Mis inversiones**

***Escenario 1: Pantalla “Mis inversiones”***

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Cuando** selecciona la opción “Mis inversiones”  
**Entonces** le debe abrir la pantalla donde le lista las inversiones que tiene el usuario con la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Mis inversiones |
| Icono | Icono definido |
| Información de la inversión | * Valor * Días para cobro * Fecha de compra |
| Botón | “Ver Detalle” Cuando el cliente le de tap deberá ir a la pantalla del detalle de la inversión. |
| Botón | Back   * Cuando el cliente la da Tap, debe dejarlo en el home del inversionista. |

***Escenario 2: Mensaje cuando el usuario no tiene inversiones***

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Y** el usuario no tiene inversiones   
**Cuando** selecciona la opción “Mis inversiones”  
**Entonces** le debe mostrar un mensaje “Aun no tienes inversiones”

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como Inversionista  
**Y** va al Home del inversionista  
**Y** el usuario tiene inversiones  
**Cuando** da TAP en el botón “Ver Detalle”  
**Entonces** le debe mostrar pantalla de detalle de la inversión.

**Registro Oferente**

***Escenario 1: Pantalla registró Oferente***

**Dado** un usuario que posee el app  
**Y** está en la pantalla de registro de usuario y contraseña  
**Y** se ha registrado exitosamente el usuario y la contraseña  
**Y** selecciono oferente como opción del registro  
**Cuando** de tap en el botón siguiente  
**Entonces** debe llevarlo a la pantalla de Registro de Oferente que tiene los siguientes componentes:

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | “Registro Oferente” |
| Nombre y Apellidos de contacto | Precargados No editable |
| Nombre de la empresa | Texto |
| Tipo de identificación de la empresa | Dropdown list con los tipos de identificación para empresa   * Campo requerido |
| Número de identificación de la empresa | Debe permitir máximo 30 caracteres.   * Campo requerido |
| Número de celular | Debe permitir máximo 15 caracteres. Solo debe permitir ingresar Números.   * Campo requerido |
| Aceptar términos y condiciones | Checkbox   * Campo requerido |
| Finalizar registro | Botón (para esta historia no tendrá funcionalidad) |

**Pantalla de Home Ofertante**

***Escenario 1: Pantalla de Menú ofertante***

**Dado** un usuario que posee el app  
**Y** se ha finalizado el registro como ofertante de manera exitosa  
**Y** hace login en la aplicación   
**Cuando** hace tap en el botón login  
**Entonces** le debe mostrar la pantalla de Home del ofertante con los siguientes campos:

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Menú Ofertante |
| Texto | “Bienvenido al home del Ofertante” |
| Icono + Texto | Icono + “Publicar factura”. Sin funcionalidad para esta historia |
| Icono + Texto | Icono + “Facturas Vendidas”. Sin funcionalidad para esta historia |
| Icono + Texto | Icono de cuenta + “Mi Cuenta”. Sin funcionalidad para esta historia. |

**Pantalla publicar factura**

***Escenario 1: Pantalla publicar factura***

**Dado** un usuario que posee la app  
**Y** ha iniciado sesión como ofertante  
**Y** va al home de ofertante  
**Cuando** da tal en la opción Publicar facturas  
**Entonces** le debe mostrar una pantalla para publicar facturas con la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Publicar Factura |
| Nombre del ofertante | * Nombre del ofertante precargado. * NO editable. |
| Valor de la factura | * Valor de la factura. |
| Moneda de la factura | * Moneda de la factura. |
| Fecha de cobro | * Formato dd/mm/aaaa. |
| Plazo de cobro | * Días. |
| Botón | * Adjuntar archivo. * Cuando el usuario haga TAP deberá abrir el explorador de carpetas y permitir adjuntar archivo. |
| Detalle/observaciones | * Texto si aplica. * No editable. |
| Botón | * Publicar oferta. * Cuando el cliente da TAP lo llevar a la historia de ofertas publicadas. |

**Pantalla de Facturas Vendidas**

***Escenario 1: Pantalla “Facturas vendidas”***

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como ofertante  
**Y** va al Home del ofertante  
**Cuando** selecciona la opción “Facturas Vendidas”  
**Entonces** le debe abrir la pantalla donde le lista las facturas vendidas que tiene el usuario con la siguiente información

|  |  |
| --- | --- |
| ***Campo*** | ***Características*** |
| Titulo | Facturas Vendidas |
| Icono | Icono definido |
| Información de la inversión | * Valor * Fecha factura * Fecha de compra |
| Botón | “Ver Detalle” Cuando el cliente le de tap deberá ir a la pantalla del detalle de la factura |
| Botón | Back   * Cuando el cliente la da Tap, debe dejarlo en el home del inversionista. |

***Escenario 2: Mensaje cuando el usuario no tiene facturas vendidas***

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como ofertantes  
**Y** va al Home del ofertante  
**Y** el usuario no tiene facturas vendidas  
**Cuando** selecciona la opción “Facturas Vendidas”  
**Entonces** le debe mostrar un mensaje “Aun no tienes facturas vendidas”

**Pantalla de Detalle de la factura**

**Dado** un usuario que posee la app  
**Y** se ha iniciado sesión correctamente como ofertante  
**Y** va al Home del ofertante  
**Y** el usuario tiene facturas vendidas  
**Cuando** da TAP en el botón “Ver Detalle”

**Entonces** le debe mostrar pantalla de detalle de la factura con la siguiente información.

# *2.1.4*. Flujo de interacción

# A continuación presentamos los flujos de interacción, los cuales nos permiten visualizar de manera global las interacciones que nos muestra la estructura general de la aplicación. Para una mayor comprensión se presenta el flujo de interacción por perfil de usuario.

# *2.1.4.1. Flujo de interacción del usuario inversor*

# En la siguiente imagen, se presenta el flujo de interacción del usuario inversor:

# 

# *2.1.4.1. Perfil de usuario Ofertante*

# En la siguiente imagen, se presenta el flujo de interacción del usuario ofertante:

# 

# *2.1.5. Prototipo*

# Un prototipo es un modelo del comportamiento del sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar las historias de usuario. Mediante el prototipo construimos los diferentes modelos del sistema que se diseña. La ventaja más importante que tenemos con el prototipo es que nos permite plasmar las decisiones de diseño y evaluar los modelos antes de desarrollar el producto final, en este caso nuestra app.

# A continuación, se presentan el prototipo de la aplicación:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

# *2.2. Diseño Técnico*

# *2.2.1. Arquitectura de la solución*

# La arquitectura propuesta para esta solución está basada en los siguientes lineamientos:

# Subsistemas autónomos en forma de microservicios que se colaboran e intercambian mensajes entre sí.

# Contenedores en donde cada microservicio es un contenedor.

# Cada uno de los microservicios implementa un conjunto de funciones coherentes y fuertemente relacionadas, desarrollándose de forma independiente.

# Cada microservicio tiene su propia base de datos.

# Las bases de datos serán de tipo NoSQL. En la implementación se utilizará MongoDB. No se descarta utilizar otro como Cassandra.

# La coherencia de los datos o entre las bases de datos, se logra mediante paso de eventos y/o mensajes utilizando un bus de eventos.

# Las aplicaciones clientes se comunicarán con los microservicios mediante una puerta de enlace de API.

# Comunicación asíncrona basada en eventos. El bus eventos se implementará utilizando la tecnología de infraestructura de agente de mensajería RabbitMQ.

# La aplicación cliente será una App para dispositivos Android.

# 

# *Figura 1. Arquitectura de la solución*

# Como se puede visualizar, en la figura anterior se definen 4 microservicios, los cuales procedemos a describir:

# *Registro de Inversionistas:* Es el conjunto de componentes que permite el registro y administración de los inversionistas.

# *Registro de Empresas:* Corresponde al conjunto de funciones que permiten el registro y administración de las empresas.

# *Publicación de facturas y documentos:* Corresponde al conjunto de funciones que permiten la publicación de las facturas y documentos soporte.

# *Confirming:* Corresponde al conjunto de funciones que permiten la compra y venta de las facturas. Se denomina confirming porque es el nombre con el que se denomina este servicio en el sector financiero.

# Para el desarrollo de los microservicios se utilizará Spring Boot como infraestructura ligera y el lenguaje de programación Java. No se descarta utilizar Python o JavaScript.

***3. Implementación***

# *3.1. Herramientas, APIs y Frameworks utilizados*

# En la ejecución de la fase de implementación se están utilizando diferentes herramientas en el desarrollo de la solución. Estas herramientas, APIs y Framework facilitan la construcción y resolución de problemas de manera muy rápida, además que nos proporcionan un ahorro de tiempo considerable en la instalación, configuración y puesta en marcha de los ambientes de desarrollo, pruebas y a futuro en producción.

# Estas herramientas se han clasificado de la siguiente manera:

# Sistemas Operativos: Android

# Infraestructura: Docker

# Frameworks: SprintBoot

# Lenguajes de programación: Java

# IDEs o ambientes de desarrollo: Android Studio, Sprint Tool Suite para Eclipse, Eclipse

# Base de Datos: MongoDB

# Mensajería: RabbitMQ

# Otros aplicativos: Postman, MongoDBCompass

# *3.1.2 Sistemas Operativos*

# Android es un sistema operativo móvil basado en el núcleo de Linux como modelo de desarrollo de código abierto. Se convirtió rápidamente en el sistema operativo para dispositivos móviles más utilizado en el mundo y por esta razón se deicidio realizar la aplicación cliente para este sistema operativo. La arquitectura de Android se compone de los siguientes componentes:

# *Linux Kernel:* Android utiliza el núcleo de Linux como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Adicionalmente el kernel es utilizado para los servicios base del sistema como administración de la memoria, gestión procesos, seguridad, controladores entre otros.

# *HAL:* Es una capa de abstracción que ofrece las interfaces que exponen las características y capacidades del hardware del dispositivo, como el módulo de la [cámara](https://source.android.com/devices/camera/index.html?hl=es-419) o [bluetooth](https://source.android.com/devices/bluetooth.html?hl=es-419), al framework de la API de Java de nivel más alto.

# *Android Runtime:* Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del [tiempo de ejecución de Android (ART)](https://source.android.com/devices/tech/dalvik/index.html?hl=es-419). El ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo.

# Arquitectura de la plataforma | Desarrolladores de Android

# *Bibliotecas C/C++ nativas:* La mayoría de los componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma proporciona API de marco de trabajo de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps.

# *Java API Framework:* Esta capa representa la base para la creación de apps Android simplificando la reutilización de componentes del sistema y de servicios. Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java.

# *Apps del sistema:* Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos.

# *3.1.1 Infraestructura*

# En infraestructura he clasificado aquellas aplicaciones y servicios que nos facilitan el montaje de un servidor o una máquina virtual. En nuestra solución estamos trabajando con plataforma de contenedores *Docker*, la cual nos proporciona un nivel de abstracción y automatización de aplicaciones en diferentes sistemas operativos. Docker es la herramienta elegida en este proyecto debido a sus prestaciones, robustez y popularidad. La arquitectura que ofrece Docker permite desplegar y ejecutar múltiples aplicaciones y soluciones sobre un mismo servidor físico, tal como se puede visualizar en la gráfica a continuación:

# Vulnerabilidades en Contenedores. Introducción | by everis Latam | Medium

# En Docker desplegamos todas las plataformas de software: base de datos, mensajería y microservicios desarrollados con sus respectivas dependencias. Docker es un software libre y de código abierto. Los contenedores Docker que desplegamos, para nuestra solución, son los siguientes:

# MongoDB

# RabbitMQ

# Servicios de Autenticación

# Servicios de Registro

# Servicios de Publicación

# Servicios de Búsqueda

# *3.1.3 Frameworks*

# Java ha sido uno de los lenguajes de programación más utilizados a la hora de desarrollar aplicaciones empresariales de cierta complejidad. En las últimas décadas se han ido creando diversos marcos de trabajo estándar (*frameworks*), los cuales se utilizan como base para desarrollar aplicaciones y soluciones en java. Uno de estos frameworks y de los más conocidos de código abierto es *Sprint*. Sprint reduce notablemente la complejidad de la especificación estándar de J2EE y del modelo de componente Enterprise Java Beans EJB. *SprintBoot* es un complemento del framework de Sprint y su principal ventaja es que agiliza y simplifica aún más la configuración de los proyectos en Sprint. SprintBoot es un marco ideal para desarrollar una aplicación o solución desde cero.

# Spring Boot proporciona la estructura básica de configuración para un proyecto en java e incluye las pautas para usar el marco y todas las bibliotecas de terceros relevantes para la aplicación, lo que permite al equipo desarrollar lo más rápidamente posible sin preocuparse tanto por los temas de configuración y parametrización de otros componentes como pueden ser un AplicationServer o Base de Datos.

Las características de Spring Boot pueden resumirse de la siguiente manera:

* Incorporación directa de aplicaciones de servidores web/contenedores como Apache Tomcat o Jetty, eliminando la necesidad de incluir archivos WAR (Web Application Archive).
* Simplificación de la configuración de Maven gracias a los POM (Project Object Models) “starter”.
* Configuración automática de Spring en la medida de lo posible.
* Características no funcionales, como métricas o configuraciones externalizadas.

Gracias a las características descritas y a su facilidad se decidió crear los microservicios de la solución con este framework.

# *3.1.4 Lenguajes de programación*

# En este TFM sobresale Java como lenguaje de programación tanto en el backend como en el front (app móvil). Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Una de las principales características de java consiste en que las aplicaciones de Java son [compiladas](https://es.wikipedia.org/wiki/Compilador) a [bytecode](https://es.wikipedia.org/wiki/Bytecode_Java) (clase Java), que puede ejecutarse en cualquier [máquina virtual Java](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java) (JVM) sin importar la [arquitectura de la computadora](https://es.wikipedia.org/wiki/Arquitectura_de_computadoras) subyacente.

El lenguaje Java se creó con cinco objetivos principales:

* Debería usar el paradigma de la programación orientada a objetos.
* Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
* Debería incluir por defecto soporte para trabajo en red.
* Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
* Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

# Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet.

# En el desarrollo de la solución se puede evidenciar que hay una fuerte tendencia a utilizar herramientas y frameworks de código abierto y/o cuya base o lenguaje de programación sea java. Esta tendencia se debe a que he querido volver a utilizar este lenguaje de programación y este TFM me abrió el espacio para hacerlo. Adicionalmente, la curiosidad de trabajar frameworks como SprintBoot y el desarrollo de aplicaciones móviles en Android donde Java ha sido el rey como lenguaje de programación.

# *3.1.5 IDE o ambientes de desarrollo*

# *3.1.5.1 Android Studio*

# Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android, basado en [IntelliJ IDEA](https://www.jetbrains.com/idea/). Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece muchas funciones que aumentan la productividad cuando se desarrollan apps para Android, como las siguientes:

* Un sistema de compilación flexible basado en Gradle.
* Un emulador rápido y cargado de funciones.
* Un entorno unificado donde puedes desarrollar para todos los dispositivos Android.
* Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación.
* Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra.
* Variedad de marcos de trabajo y herramientas de prueba.
* Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros.
* Compatibilidad con C++ y NDK.
* Compatibilidad integrada con [Google Cloud Platform](https://cloud.google.com/tools/android-studio/docs/?hl=es-419), que facilita la integración con Google Cloud Messaging y App Engine.

# *3.1.5.2 Eclipse*

# Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Esta plataforma ha sido utilizada para desarrollar entornos de desarrollo integrados como el IDE de java denominado Java Development Toolkit (JDK) y el compilador (ECJ) que se instala como parte de eclipse.

# El [entorno de desarrollo integrado](https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) (IDE) de Eclipse emplea módulos (*plug-in*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software.

La base para Eclipse es la Plataforma de cliente enriquecido ([Rich Client Platform](https://es.wikipedia.org/w/index.php?title=Rich_Client_Platform&action=edit&redlink=1" \o "Rich Client Platform (aún no redactado)) RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

* Plataforma principal - inicio de Eclipse, ejecución de plugins.
* [OSGi](https://es.wikipedia.org/wiki/OSGi) - una plataforma para bundling estándar.
* El [Standard Widget Toolkit](https://es.wikipedia.org/wiki/SWT) (SWT) - Un [widget toolkit](https://es.wikipedia.org/w/index.php?title=Widget_toolkit&action=edit&redlink=1) portable.
* [JFace](https://es.wikipedia.org/wiki/JFace) - manejo de archivos, manejo de texto, editores de texto.
* El Workbench de Eclipse - vistas, editores, perspectivas, asistentes.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

# *3.1.5.3 Sprint Tool Suite - Eclipse*

# SpringSource Tool Suite (STS) es un IDE basado en la versión Java EE de Eclipse, pero altamente customizado para trabajar con Spring Framework. Entre las características más destacadas que STS proporciona se encuentran:

* Soporte para Spring 3.
* Asistentes para la creación de proyectos Spring.
* Herramientas para la gestión de beans.
* Editores gráficos de archivos de configuración de Spring.
* Herramientas de desarrollo para Spring Web Flow y Spring Batch.

STS puede [descargarse como un IDE completo](http://www.springsource.com/landing/best-development-tool-enterprise-java) o integrarse sobre una instalación previa de Eclipse.

# *3.1.6 Base de Datos*

Para el desarrollo de la solución se ha decidió trabajar con el motor de base de datos NoSQL Mongo DB. MongoDB es un sistema de [base de datos](https://es.wikipedia.org/wiki/Base_de_datos) [NoSQL](https://es.wikipedia.org/wiki/NoSQL), [orientado a documentos](https://es.wikipedia.org/wiki/Base_de_datos_documental) y de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto).

En lugar de guardar los datos en tablas, tal y como se hace en las [bases de datos relacionales](https://es.wikipedia.org/wiki/Base_de_datos_relacional), MongoDB guarda estructuras de datos [BSON](https://es.wikipedia.org/wiki/BSON) (una especificación similar a [JSON](https://es.wikipedia.org/wiki/JSON)) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. El modelo de documento se asigna a los objetos en el código de la aplicación para facilitar el trabajo de los datos.

# *3.1.7 Mensajería*

# En el TFM se ha incorporado el trabajar con un software de mensajería con el fin de utilizarlo para el manejo de mensajes asíncronos en diferentes instantes o funcionalidades de la solución. El software seleccionado es *RabbitMQ*. RabbitMQ es un software de negociación de mensajes de código abierto que funciona como un [middleware](https://es.wikipedia.org/wiki/Middleware) de mensajería. Implementa el estándar [Advanced Message Queuing Protocol](https://es.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) ([AMQP](https://es.wikipedia.org/wiki/AMQP)). El servidor RabbitMQ está escrito en [Erlang](https://es.wikipedia.org/wiki/Erlang) y utiliza el *[framework](https://es.wikipedia.org/wiki/Framework)* [Open Telecom Platform](https://es.wikipedia.org/w/index.php?title=Open_Telecom_Platform&action=edit&redlink=1) (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores.

El proyecto RabbitMQ consta de diferentes partes:

* El servidor de intercambio RabbitMQ, que recibe los mesajes y los almacena en colas de intercambio. Los clientes podrán después recuperarlos.
* Pasarelas para los protocolos [HTTP](https://es.wikipedia.org/wiki/HTTP), [XMPP](https://es.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol) y [STOMP](https://es.wikipedia.org/w/index.php?title=STOMP&action=edit&redlink=1).
* Bibliotecas de clientes para [Java](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)), *[framework](https://es.wikipedia.org/wiki/Framework)* [.NET](https://es.wikipedia.org/wiki/.NET) y [Python](https://es.wikipedia.org/wiki/Python).
* El [plugin](https://es.wikipedia.org/wiki/Plugin) [Shovel](https://es.wikipedia.org/w/index.php?title=Shovel&action=edit&redlink=1" \o "Shovel (aún no redactado)) (en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s): [pala](https://es.wikipedia.org/wiki/Pala)) que se encarga de replicar mensajes entre corredores de mensajes.

# *3.1.8 Otras herramientas*

# Adicional a las herramientas ya descritas, estamos utilizando la herramienta *Postman* para hacer pruebas de los servicios desarrollados y MongoDB Compass para visulizar las colecciones y documentos en la base de datos mongodb.

**Postman** nace como una herramienta que principalmente nos permite crear peticiones sobre APIs de una forma muy sencilla y así probar las APIs. Los usuarios de **Postman** pueden ser:

* Un desarrollador que esté comprobando el funcionamiento de una API para desarrollar sobre ella
* Un operador el cual esté realizando tareas de monitorización sobre un API

**Postman** nos ofrece un conjunto de utilidades adicionales para poder gestionar las APIs de una forma más sencilla.

***MongoDB Compass*** analiza los documentos y, a través de su intuitiva interfaz gráfica de usuario, le muestra estructuras detalladas en sus colecciones. De este modo, puede visualizar y explorar rápidamente el esquema para comprender la frecuencia, tipo y rango de campos del conjunto de datos.

# *3.2. Aspectos Relevantes del desarrollo*

# *3.2.1 Desarrollo de App Android*

# La aplicación cliente o app móvil se desarrolla en Android Studio 4.1 de forma nativa y será compatible con los dispositivos que cuenten con la versión Android 5.0 (lollipop) en adelante, es decir será compatible con el 94.1% de los dispositivos móviles Android.

# La estructura del proyecto es la siguiente:

# En la carpeta java se encuentran las clases del proyecto de manera estructurada por paquetes. En el paquete *com.caam.confirming.interface* se encuentran las interfaces creadas para el consumo de los microservicios. En el paquete *com.caam.confirming.models* se encuentran las clases que representan cada uno de los modelos de objetos que intervienen en la aplicación. Aquí encontramos clases como Inversor, Ofertante y Login. Adicionalmente, se encuentran clases que representan o definen el objeto que retorna cada uno de los servicios. En el paquete *com.caam.confirming.ui* se encuentra el conjunto de clases que se asocia a cada uno de los layout o activities.

# 

# En la carpeta *res/layout* se encuentran los layout de cada activity creada. Recordemos que los activities es el conjunto de pantallas (screens) que visualizara el usuario de la app en su dispositivo.

# 

# En la carpeta *Gradle Scripts*, script *build.grade* se referencian las dependencias que se utilizarán en la app. Hemos adicionado las dependencias:

# 

# *Retrofit2* son el conjunto de bibliotecas de redes más utilizadas en Android, es decir las utilizamos para facilitar la invocación a los microservicios como se puede ver a continuación:

# 

# Junto con *retrofit2* en el script *build.gradle* se describe las librerías *Gson* las cuales permiten la serialización y deserialización entre objetos java y su representación en notación *JSON*. Los microservicios retornan o response JSON object y con el uso de Gson nos permite serializarlos a objetos/clases java, tal como puede visualizar a continuación:

# 

Cuando se crea un activity, Android Studio brinda la oportunidad de generar esta activity con una plantilla. La activity inicial que corresponde al *activity\_login.xml*, se creó con la plantilla predefinida Login Activity. Las demás activities se han creado con la plantilla *Empty Activity* la cual nos provee una plantilla libre para trabajar.

# En cada layout se ha trabajado con el diseño *LinearLayout* que corresponde a un grupo de vista que alinea todos los componentes secundarios en un única dirección, de manera vertical u horizontal, de acuerdo con el valor que se defina el atributo *android:orientation.*

# 

# *3.2.2 Desarrollo de los microservicios SprintBoot*

# Para el desarrollo de los microservicios se utilizó *Sprint Tool Suite 4* que se basa en el ecosistema de eclipse facilitando la creación y desarrollo de los servicios bajo el framework de Sprint y su evolución SprintBoot.

# El servicio se crea como un proyecto tipo *Sprint Starter Project* en donde un wizard guía al desarrollador indicando la versión del jdk, dependencias requeridas y demás capacidades o características que va a tener el servicio.

# Los principales aspectos de configuración a tener en cuenta y revisar una vez se crea el proyecto son:

# *Archivo pom.xml:* Este archivo se encuentra en la raíz del proyecto y corresponde a las siglas Project Object Model el cual es la unidad básica y principal de un proyecto Maven. Este archivo contiene información acerca del proyecto, las fuentes, dependencias, plugins etc. Como dependencias para nuestro desarrollaremos utilizamos las siguientes:

# 

# Como se puede visualizar en la imagen anterior utilizamos dependencias para conectarse con mongodb, así como rabbitmq. En este archivo también se define la versión de java con la que se compilará y ejecutará el servicio, en nuestro caso es java 11.

# 

# Cada uno de los proyectos puede contener 1 o 2 microservicios y cada proyecto tiene su archivo *pom.xml*

# En la carpeta *src/main/resources* se encuentra el archivo *application.properties*. En este archivo se definen parámetros para conexión con mongodb y definición del puerto en el cual se desea subir o ejecutar el servicio. Recordemos que SprintBoot incluye o tiene embebido Tomcat. A continuación, se presenta un ejemplo de este archivo.

# 

# En la carpeta *src/main/java* se crea un paquete el cual contiene una clase java con el método *public static void main(String[] args)*. El nombre de este paquete se define en el wizard de creación del proyecto. Esta clase es la ejecutamos con *Run -> Java Application*, es decir una de las bondades de SprintBoot es que puedes subir y publicar un servicio REST como si fuese una aplicación de consola. A continuación, se presenta una imagen de cómo se visualiza esta ejecución en el ambiente de desarrollo.

# 

# Para nuestro desarrollo hemos creado los paquetes *.model*, *.repository* y *.resources* como estructura para mantener nuestras clases. La estructura de un proyecto se visualiza a continuación:

# 

# Tomando como base la imagen anterior, el paquete *com.confirming.reg.ofertante.model* se encuentran las clases que representan cada uno de los modelos de objetos que intervienen en la solución. En el paquete *com.confirming.reg.ofertante.repsitory* se definen las interfaces que extienden de *MongoRepository* y con las cuales se crean las conexiones con mongodb. En el paquete *com.confirming.reg.ofertante.resources* se encuentran las clases tipo controller mediante la notación *@RestController* como se puede visualizar a continuación.

# 

# Para la conexión con RabittMQ se define el paquete *.config* en donde la clase *MessaginConfig.class*, define los parámetros de configuración con Rabbit y en el paquete *.publisher* definimos el servicio que envía el mensaje a Rabbit.

|  |  |
| --- | --- |
|  |  |

# Una vez tenemos nuestros servicios creados podemos llevarlos a un contenedor. Para esto creamos un archivo *Dockerfile* carpeta raíz del proyecto como se visualiza a continuación:

# 

# *3.3. Pruebas*

# Con el fin de garantizar un funcionamiento correcto de la solución se realizan una serie de pruebas a la aplicación en la fase de desarrollo. Las pruebas que se han realizado son las pruebas unitarias con el fin de comprobar que cada parte del código funciona de acuerdo con lo esperado.

# Hemos utilizado el utilitario Postman para realizar las pruebas sobre los microservicios. A continuación, describimos las pruebas realizadas:

|  |  |
| --- | --- |
| *Nombre de la Prueba* | Test microservicio de autenticación |
| *Descripción* | Este microservicio recibe como parámetro 2 valores, el login y la contraseña. Retornando un JSON con 3 valores con la siguiente estructura: {      "isUser": "",      "tipoUsuario": "",      "nombre": ""  } |
| *Resultado esperado* | Cuando el usuario NO existe debe retornar el siguiente objeto JSON: {      "isUser": "",      "tipoUsuario": "",      "nombre": ""  }  Cuando el usuario existe debe retornar el objeto con datos:  {      "isUser": "true",      "tipoUsuario": "inversor",      "nombre": "Luis"  } |

# Se realiza la prueba utilizando Postman, con datos que no existen en la base de datos. El resultado es:

# 

# Se realiza la prueba utilizando Postman, con datos de usuario ya registrado en la base de datos. El resultado es:

# 

|  |  |
| --- | --- |
| *Nombre de la Prueba* | Test microservicio registro Inversor |
| *Descripción* | Este microservicio recibe un objeto JSON con la siguiente estructura: {      "id":79719360,  "nombre":"Carlos",      "apellido":"Agudelo",      "email":"carlosag@gmail.com",      "contrasena":"CarlosAg2020",      "tipoIdentificacion":"cedula de ciudadania",      "identificacion":"79719360",      "moneda":"USD",      "monto":"28000",      "telefono":"3164727725"  } |
| *Resultado esperado* | Cuando el usuario se registra correctamente debe retornar el siguiente objeto JSON: {      "insertR": "Ok",      "error": "no error",  } |

# Se realiza la prueba utilizando Postman:

# 

|  |  |
| --- | --- |
| *Nombre de la Prueba* | Test microservicio registro Login |
| *Descripción* | Este microservicio recibe un objeto JSON con la siguiente estructura: {      "id":79719360,      "login":"carlosag@gmail.com",      "contrasena":"CarlosAg2020",      "tipoUsuario":"inversor",  } |
| *Resultado esperado* | Cuando el usuario se registra correctamente debe retornar el siguiente objeto JSON: {      "insertR": "Ok",      "error": "Sucess !!",  } |

# Se realiza la prueba utilizando Postman:

# 

|  |  |
| --- | --- |
| *Nombre de la Prueba* | Test microservicio registro Ofertante |
| *Descripción* | Este microservicio recibe un objeto JSON con la siguiente estructura: {      "id":1002564555,      "nombre":"Salome",      "apellido":"Moncada",      "email":"salo@gmail.com",      "contraseña":"salo1943",      "empresa":"Salome Society",      "tipoIdentificacionEmpresa":"nit",      "identificacionEmpresa":"1002564555",      "telefono":"31610000"  } |
| *Resultado esperado* | Cuando el usuario se registra correctamente debe retornar el siguiente objeto JSON: {      "insertR": "Ok",      "error": "no error",  } |

# Se realiza la prueba utilizando Postman:

# 

# *3.4. Funcionalidad de la aplicación:*

# A continuación, se presentan las capturas de las pantallas de la aplicación en donde se puede observar las funcionalidades:

# *3.4.1 Login y Registro de Usuarios*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

# *3.4.2 Menú de Usuarios*

|  |  |
| --- | --- |
|  |  |

# *3.4.3 Búsqueda y publicación*

|  |  |  |
| --- | --- | --- |
|  |  |  |

# *Pendientes de la implementación:*

# A continuación, se describen los pendientes a la fecha para finalizar la implementación:

# Se debe finalizar la integración de la aplicación móvil con los microservicios desarrollados.

# Documentación de las pruebas unitarias de algunos microservicios ya desarrollados.

# Pruebas de integración con la app móvil.

# Mejorar el *look & feel* de la app móvil.

# *Bibliografía*

# *Kubernetes Microservices with Docker*

# *Autores: Deepak Vohra*

# *2016*

# *Microservices from Day One: Build robust and scalable software from the start*

# *Autores: Cloves Carneiro Jr.*

# *Tim Schmelmer*

# *2016*

# *Learn Microservices with Spring Boot: A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber*

# *Autores: Moises Macero*

# *2017*

# *Building Event-Driven Microservices*

# *Autores: Adam Bellemare*

# *2020*

## ***Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement***

## *Autores: Eric Redmond*

## *Jim R. Wilson*

## *May 2012*

# *Microservicios de .NET: Arquitectura para aplicaciones .NET en contenedor*

*Autores: Cesar de la Torre*

*Bill Wagner*

*Mike Rousos*

<https://docs.microsoft.com/es-es/dotnet/architecture/microservices/>

***Edición v3.1.2***

# *Orquestación de aplicaciones corporativas mediante virtualización de contenedores usando Docker*

# *Autores: José Angel Fernández Ameijeiras*

# *Grado de Ingeniería informática*

# *José Manuel Castillo Pedrosa*

# <https://www.rabbitmq.com/>

# <https://spring.io/projects/spring-boot>

# <https://www.mongodb.com/es>

# <https://www.finnovating.com/news/por-que-fintech-crece-tanto-y-tan-rapido-en-latinoamerica/#:~:text=El%20futuro%20de%20Fintech%20en,millones%20de%20d%C3%B3lares%20en%202021>

# <https://www.larepublica.co/finanzas/cifras-de-operaciones-y-movilizacion-de-dinero-en-internet-y-banca-movil-en-colombia-3042211>