UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Santa Fe

PR

PedíRápido

Plataforma web de gestión de pedidos con envío a domicilio

PROYECTO FINAL DE CARRERA

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

- AÑO 2016 -

ALUMNOS

GUICHARD, Santiago;
OÑATE, Germán

DIRECTOR

Ing. VRANCKEN, Lisandro

Agradecimientos

A nuestra facultad, por brindarnos no solo conocimientos, sino la capacidad de enfrentar todo tipo de problemas. Y a nuestros familiares y amigos, por apoyarnos, estar a nuestro lado durante todo este camino y hacer posible este gran sueño.

Índice de contenidos

Índice de figuras

Índice de tablas

Organización del informe

El informe final del presente proyecto se encuentra dividido en secciones para una mayor comprensión, cada una de las cuales se detalla a continuación:

SECCIÓN 1, "Introducción": en esta sección se plantean el tema, el problema detectado, los objetivos y su fundamentación teórica. Nos brinda un marco conceptual para comprender la idea y ejecución del proyecto.

SECCIÓN 2, "Desarrollo del proyecto": aquí se presentan las tres etapas que aplicamos durante el proyecto (inicial, iterativa y final), así como también los procedimientos y sub etapas que conforman cada una de ellas.

SECCIÓN 3, "Arquitectura definida y requerimientos no funcionales": contiene el documento de arquitectura inicial del proyecto y los principales requerimientos no funcionales que lo conforman.

SECCIÓN 4, "Tecnologías y entorno de desarrollo": se identifican las principales herramientas y piezas de software utilizadas para llevar a cabo el proyecto.

SECCIÓN 5, "Trabajos futuros": se plantea la situación actual del proyecto y qué trabajos y extensiones se pueden y se esperan realizar en el futuro.

SECCIÓN 6, "Conclusiones": presenta las conclusiones de la planificación realizada y del trabajo realizado como parte de este proyecto.

SECCIÓN 7, "Anexo": contiene el manual de instalación, resultados de pruebas funcionales y de performance, incluyendo además una galería de imágenes.

Por último se incluye una sección “Glosario” donde se definen palabras y expresiones técnicas utilizadas, y también una sección denominada “Referencias y bibliografías” donde brindamos información que posibilita identificar libros, publicaciones y sitios web que utilizamos para el desarrollo del proyecto y del informe.

1. INTRODUCCIÓN

a. Temática del proyecto

Este proyecto final de carrera fue llevado a cabo con la finalidad de construir un conjunto de aplicaciones web que permitan dar soporte a los procesos relacionados con la gestión de pedidos a domicilio, focalizándose en los siguientes rubros: comidas, helados y bebidas

b. Problema a resolver

El avance tecnológico ha llegado a todos los rincones del mundo generando nuevas aplicaciones utilizadas en diferentes ámbitos. Las mismas son accesibles desde distintos dispositivos ayudando a tener mayor información de manera más ordenada y más accesible.

A nivel nacional, un gran número de empresas, existentes y nuevas, envían sus productos a domicilio y otras no lo hacen debido al costo o al trabajo adicional que ello requiere. A su vez, los consumidores en cada ocasión que requieren realizar algún pedido no poseen un medio centralizado de información, que abarque distintos rubros.

El producto que construimos está orientado a dar soporte a los procesos involucrados en la gestión de pedidos y envíos a domicilios, a través del desarrollo de aplicaciones web con distintos fines y tipos de usuarios.

La característica principal que diferencia nuestro servicio del resto, se encuentra en que no nos concentramos únicamente en los clientes que realizan los pedidos, sino que ofrecemos una herramienta a los comerciantes que permite simplificar y unificar la gestión de su negocio.

Nuestro servicio pretende alentar a los nuevos y pequeños comerciantes a poner el nombre de su negocio frente a una inmensa cartera de clientes, y afianzar a los comerciantes con más trayectoria generando publicidad y ofreciendo sus productos delante de clientes que pueden acceder a los mismos desde cualquier lugar y desde cualquier dispositivo.

c. Objetivos

i. Objetivo general:

Desarrollar una aplicación web para que usuarios puedan realizar pedidos online a distintos negocios. Y que estos, a su vez, puedan gestionar todos los pedidos que reciben.

ii. Objetivos específicos:

- Disponer de información fehaciente del mercado sobre las necesidades del negocio de pedidos y envíos a domicilio.
- Crear y desplegar una aplicación web que permita realizar pedidos a un cliente final.
- Crear y desplegar una aplicación web para permitir al cliente proveedor gestionar los pedidos.
- Crear y desplegar una aplicación web para la administración del sistema.

d. Ámbito y alcance

El desarrollo de este proyecto se llevó a cabo en la Universidad Tecnológica Nacional, Facultad Regional Santa Fe, durante el período octubre 2015 a septiembre 2016.

Se proyecta implantar el producto final en las ciudades de Santa Fe, Paraná y Concordia, y una vez consolidado en dichas ciudades se planea la expansión a las principales ciudades de la región.

Se espera que puedan hacer uso del sistema todas aquellas personas, a los cuales llamaremos “clientes finales”, que residan en ciudades donde la aplicación esté activa y tengan acceso a internet a través de cualquier dispositivo. Por otra parte, se pretende el uso por parte de aquellos comercios que quieran ofrecer sus pedidos en nuestra aplicación (“clientes proveedores”). El producto estará especialmente orientado a personas jóvenes con afinidad a esta tecnología, aunque se considerarán los atributos de accesibilidad y usabilidad que permitan ampliar el uso por parte de usuarios que no reúnan estas características.

Los usuarios podrán:

- En base a la dirección y el rubro, buscar negocios que se podrán ordenar en base a diferentes criterios.
- Elegir productos, promociones y ver información asociada al negocio.
- Una vez realizado el pedido, seguir el estado del mismo.

Los negocios:

- Administrar de forma centralizada sus pedidos, con la posibilidad de ir cambiándolos de estado.
- Agregar, modificar y dar de baja los productos que ofrecen.

e. Marco teórico

El concepto de "e-commerce"[1] consiste en la distribución, venta, compra, marketing y suministro de información de productos o servicios a través de Internet. Considerando de conocimiento público la tendencia mundial del uso de aplicaciones web y móviles para distintos rubros, en especial en el ámbito del retail moderno, toma relevancia el desarrollo del proyecto sobre este dominio. El crecimiento exponencial y expansión de este tipo de aplicaciones, se debe entre otros motivos a la facilidad, comodidad, rapidez y aumento de la información que adquiere el usuario a la hora de tomar una decisión. Las nuevas tecnologías disponibles ofrecen variedad de posibilidades para lograr un servicio innovador, un diseño exclusivo adaptado al usuario en todas aquellas plataformas de mayor aceptación.

Se exponen a continuación los factores que favorecen el crecimiento, según estudio anual del comercio electrónico del CACE[2]:

- El continuo crecimiento del número total de usuarios de Internet en el país. De 7.6 millones de usuarios a fines de 2004 a 34.5 millones a fines de 2015.
- n 2015 fue de $68.5 millones de pesos (crecimiento del 70.8% respecto al 2014).
- El aumento de la proporción de usuarios de internet que realizaron compras on-line: 10% aproximado en 2001 al 77% en 2015 (superando los 17 millones de personas).
- Fortalecimiento del *mobile commerce*: el 36% del tráfico total en comercio electrónico en Argentina provino de dispositivos móviles.
- Mejores formas de pago y promociones.

[1] Fuente: http://marketingdigital.bsm.upf.edu/e-commerce-comercio-electronico/
[2] CACE (Cámara Argentina de Comercio Electrónico) http://www.cace.org.ar/estadisticas/

- Las transacciones recurrentes. El 52% de las personas que realizaron compras por internet son compradores recurrentes, contra un 37% del año 2014.

Durante el transcurso de los últimos años el uso de navegadores en su versión desktop fue decreciendo dando lugar a las versiones móviles. Es por eso, que hoy en día la tendencia es implementar un diseño responsivo, tanto por las ventajas que brinda como también por el crecimiento antes mencionado.

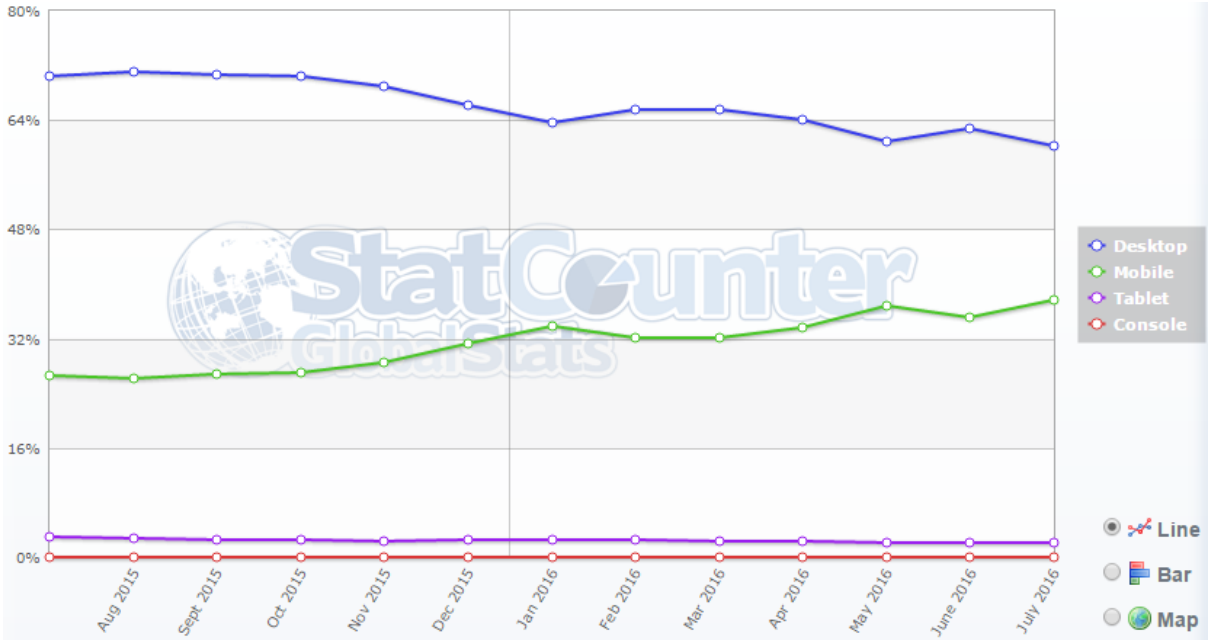

Figura 1-1 Cuadro comparativo de acceso a navegadores desde distintos dispositivos

Beneficios del diseño adaptativo:

- Mejora la experiencia de navegación del usuario: el usuario que accede a una web a través de un móvil y no puede percibir el contenido correctamente porque éste no se encuentra encuadrado, probablemente no vuelva a entrar a la web.
- Contenido duplicado: antes se tenía el mismo contenido para una versión adaptada al móvil y otro igual adaptado a otros dispositivos.
- Mejor visibilidad: Google no tardó en darse cuenta del creciente uso de internet en los dispositivos móviles, por esto, otorga un lugar destacado en los motores de búsqueda para aquellas webs optimizadas para móviles. El último cambio en el algoritmo del buscador ha beneficiado a todas las páginas web *mobile-friendly*, otorgándoles un mejor posicionamiento sobre aquellas webs que aún no están optimizadas.
- Ahorro de tiempo y costo: dejando de lado tener que diseñar y desarrollar una versión móvil para nuestra web y, por lo tanto, el costo que esto lleva. Como así también el tiempo de mantener dos versiones.

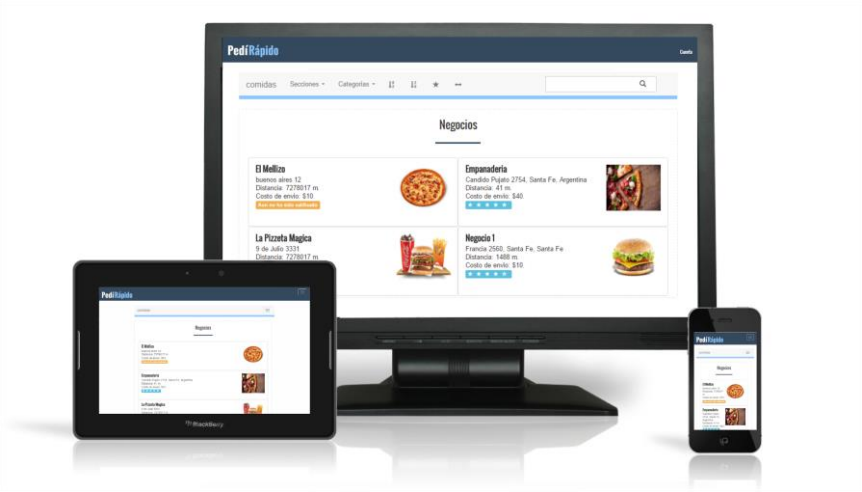

Figura 1-2 PediRapido desde distintos dispositivos

2. DESARROLLO DEL PROYECTO

En esta sección se describe en detalle el proceso de realización de este proyecto. El cual dividimos en tres etapas. En la primera, denominada “etapa inicial”, se definieron las funcionalidades, metodología de trabajo y realizamos el análisis y diseño iniciales del proyecto. Luego, en la “etapa iterativa”, se implementaron las funcionalidades especificadas. Y por último, en la “etapa final”, realizamos las pruebas establecidas, para asegurar la calidad del sistema, para posteriormente realizar su instalación.

a. Etapa inicial

i. Definición de las funcionalidades del sistema

A continuación, se describen las principales funcionalidades que se han desarrollado, agrupadas por el usuario que las lleva a cabo:

Cliente final:

- **Alta, baja y modificación usuario:** permite crear, modificar y dar de baja una cuenta de usuario final, el cual realiza los pedidos dentro de la aplicación.
- **Detección automática de ubicación:** la misma brinda la posibilidad de detectar automáticamente la ubicación del usuario, además de poder ingresarla manualmente o recordar la última utilizada.
- **Elección de rubro:** el usuario puede elegir dentro de una lista de rubros para filtrar los negocios ofrecidos por la aplicación.
- **Diferentes criterios de búsqueda:** el usuario puede filtrar los negocios por diferentes criterios, como son: distancia, orden alfabético, popularidad y tipo producto.
- **Armar y confirmar pedidos para un local:** una vez seleccionado el local, el usuario arma su pedido eligiendo productos agrupados por categorías e indicando la cantidad de unidades, luego confirma el mismo.
- **Seguimiento del estado del pedido:** luego de que el pedido es tomado por parte del negocio que lo ofrece, el usuario podrá ver desde su historial de pedidos el estado del mismo.

Cliente proveedor:

- **Alta, baja y modificación comercio:** permite crear, modificar y dar de baja un usuario proveedor, el mismo es solicitado por el cliente proveedor pero solo tiene acceso a modificar sus datos.
- **Gestión de productos ofrecidos y promociones:** el usuario puede modificar disponibilidad de determinados productos o promociones, dar de alta nuevos productos y promociones y modificar sus precios.
- **Gestión de pedidos:** el usuario cuenta con una interfaz donde se muestra una lista con los pedidos solicitados según sus estados y puede ejecutar acciones asociadas a los mismos, como aceptarlos, rechazarlos o indicar que se están enviando.

Administración:

- **Alta, baja y modificación Administrador:** permite crear, modificar y dar de baja un usuario administrador.
- **Administración usuarios, negocios y proveedores:** el usuario cuenta con una interfaz desde donde puede realizar altas, bajas y modificaciones del resto de los tipos de usuarios definidos.
- **Cobranzas:** el usuario accede a una lista desde donde puede administrar los pagos de todos los negocios.

ii. Definición de la metodología

A continuación se describe la metodología de trabajo que llevamos a cabo para el desarrollo del presente proyecto, características generales y específicas y justificación de la elección e implementación de la misma.

Decidimos utilizar distintas prácticas de metodologías ágiles, basándonos principalmente en la metodología XP (programación extrema). Esta elección se debió a los siguientes motivos:

- Dada la falta de experiencia en este tipo de proyectos, creemos que, si definimos un plan estructurado, se dificultará la adaptación de los cambios que surjan durante el mismo, cuestión que se puede manejar mejor con metodologías ágiles.
- Como somos un número reducido de personas, consideramos que las actividades a realizar se pueden asignar más dinámicamente que de la forma en que se asignan en un plan prescriptivo.
- Debido a las características del ámbito del proyecto, de constante cambios, alta competencia y períodos cortos, creemos que la filosofía de las metodologías ágiles, de darle prioridad a entregas de software de valor sobre la documentación, se ajusta de mejor manera.

Características de la metodología:

Basados en métodos ágiles, utilizamos historias de usuario que describen de manera general los requisitos funcionales del sistema, luego definimos la duración de las iteraciones, la cantidad y qué historias las forman, basándose en esfuerzo y la prioridad de cada historia.

¿Cómo implementamos XP?:

Describimos, a continuación, cómo llevamos a cabo los principales puntos característicos de la metodología en la que nos basamos, XP:

- Cliente como parte de equipo: al no poseer un cliente bien definido, implementamos el rol de dueño del producto, el cual estaba formado por los miembros del equipo y también, al final de cada iteración, por el director de proyecto. El objetivo fue siempre garantizar la satisfacción de los requerimientos determinados.
- Programación de a pares: no fue aplicado, al ser solo dos desarrolladores las historias eran asignadas a uno solo. Se intentó siempre que las asignaciones fueran variando en rol, función y área.
- Estándares: Se definió un estilo de diseño para toda la aplicación, que incluyó paleta de colores, fuentes, tamaños y disposición de los elementos en la pantalla. También aplicamos criterios de nombramientos en métodos, variables, atributos y base de datos, así como buenas prácticas en las que estaba basado el principal framework utilizado.
- Integración continua: por más que no utilizamos ninguna herramienta que automatice este tipo de tareas, semanalmente implantamos en nuestro servidor de desarrollo los nuevos cambios producidos hasta ese entonces. En cada una de estas implantaciones utilizamos Maven, que es una herramienta muy potente, la cual corría automáticamente un conjunto de pruebas de integración.
- Pruebas unitarias: de la mano de la herramienta nombrada en el punto anterior, también se corrían un conjunto de pruebas unitarias sobre el código fuente luego de empaquetar el mismo.
- Entregables pequeños: Se dividió el total del proyecto en 8 iteraciones, donde cada una de ellas tenía como resultado el agregado de nuevas funcionalidades que agregaban valor al producto.
- Juego de planificación: sólo los desarrolladores planificamos y estimamos las funcionalidades en base al esfuerzo, prioridad y riesgo. Esto también es explicado con mayor detalle en la sección estimaciones.

Justificación de la metodología:

Consideramos necesaria la definición de una metodología propia debido a que las demás metodologías están contempladas para desarrollos de proyectos por equipos consolidados y con experiencia, y advertimos que podemos mejorar esta situación realizando algunas modificaciones. Por eso adoptamos las mejores prácticas de las metodologías vistas durante el transcurso de la carrera para adaptarlas a nuestro proyecto.

Cada iteración estuvo formada de las siguientes etapas:

- Preparación: en esta etapa nos interiorizamos en las tecnologías utilizadas, preparamos los entornos de desarrollo y planificamos lo que íbamos a realizar en toda la iteración. Esta etapa fue aplicada en las primeras iteraciones.
- Diseño: en esta etapa modelamos los objetos relacionados con funcionalidades pertenecientes a la iteración, se realizó el diseño definitivo de las interfaces de usuario y establecimos la sintaxis de los mensajes de comunicación entre los sistemas.
- Desarrollo: en esta etapa implementamos lo planificado para la iteración.
- Prueba: en esta etapa realizamos las pruebas de unidad, integración y seguridad.
- Despliegue: implementamos la aplicación en un servidor de preproducción que simula condiciones reales de trabajo.
- Revisión de la iteración: en esta etapa realizamos el seguimiento de las tareas, evaluando qué cosas se hicieron bien o mal durante la iteración, si se respetaron los plazos establecidos, etc. A partir de las conclusiones determinamos las medidas necesarias para evitar problemas en las iteraciones sucesivas.

A modo de resultado, creemos importante destacar la experiencia en este aspecto. Consideramos que mediante la aplicación de esta metodología se obtuvieron resultados exitosos. Las principales ventajas nombradas en la descripción de la misma fueron las que aprovechamos al máximo para que los dos integrantes que formamos el equipo podamos finalizar lo que se planificó y estimó en un tiempo y forma considerablemente aceptable. Ambos integrantes del equipo de proyecto sufrimos cambios en la cantidad de horas disponibles al iniciar actividades laborales lo cual nos redujo el tiempo de trabajo en el proyecto, y gracias a la flexibilidad de la metodología pudimos compensar este tiempo con horas extras y así nunca detener el avance del proyecto.

iii. Análisis

A continuación definimos las historias de usuario y el análisis realizado en cada una de ellas, lo que denominamos estimaciones. Luego refinamos las mismas y por último definimos la velocidad del equipo.

i. Historias de usuario

Presentamos a continuación el formato utilizado para describir las historias de usuarios definidas para el desarrollo del proyecto:

Formato de Historia:
<Letra>. <nro. subsistema>. <nro. funcionalidad>
P- lado del proveedor
C- lado del cliente
A- lado del administrador

Para luego, describir las historias de usuario organizadas por las secciones establecidas:

Sección cliente final:

C.1 Inicio de sesión de usuario: el usuario ingresa a la opción “ingresar” de la barra superior y puede iniciar sesión ingresando un usuario y contraseña. También tiene las opciones de registrar una cuenta y de recuperar su contraseña.

Además puede ingresar al sitio como invitado, sin necesidad de iniciar sesión (no se guardará su actividad).

C.2 Alta Usuario: el usuario ingresa a la aplicación, y si no inició sesión tiene la opción de registrarse, se le solicitará un nombre de usuario, una dirección de email, una contraseña y una verificación de contraseña. Una vez completado el formulario de registro, el usuario recibirá un correo de habilitación para su cuenta.

C.3 Indicar ubicación:

- **C.3.1 ingreso manual:** el usuario elige ciudad de una lista e ingresa la dirección en forma de texto.
- **C.3.2 ingreso automático:** al oprimir el botón para detectar ubicación, aparece en el cuadro de texto la dirección identificada a partir de la ubicación del usuario, que se puede editar para corregir la dirección propuesta.
- **C.3.3 ingreso predeterminado:** si el usuario inició sesión, e ingresó previamente su dirección en la sección “resumen de cuenta”, esta dirección almacenada aparecerá por defecto.

C.4 Elegir rubro: el usuario podrá elegir entre una serie de rubros, que inicialmente serán “Comidas”, “Heladerías” y “Bebidas”. De esta forma sólo verá negocios correspondientes al rubro elegido.

C.5 Elegir el comercio:

- **C.5.1 Filtrar búsqueda por orden alfabético:** el usuario indica que desea ver el listado de comercios ordenado alfabéticamente.
- **C.5.2 Filtrar búsqueda por cercanía:** el usuario indica que desea ver el listado de comercios ordenado del más cercano al más lejano, a partir de la dirección indicada.
- **C.5.3 Filtrar búsqueda por tipo de comidas (sólo para la sección de comidas):** el usuario indica que desea ver el listado de comercios que tienen el tipo de comida indicado en el menú, este filtro puede ser combinado con los filtros anteriores.
- **C.5.4 Filtrar búsqueda por popularidad:** el usuario indica que desea ver el listado de comercios ordenado por un ranking de popularidad, basado en las recomendaciones.

C.6 Armar el pedido:

- **C.6.1: Elegir categoría de producto:** el usuario puede ver una lista de categorías de productos que comercio ofrece. Al seleccionar una categoría se despliega una lista con todos los productos pertenecientes a ella.
- **C.6.2: Agregar productos al pedido:** el usuario selecciona un producto y la cantidad deseada, al confirmar se agrega el producto al pedido.

C.7 Confirmar el pedido (resumen del pedido): una vez que el usuario termina de armar el pedido se muestra un resumen que muestra una lista con todo lo que se ha pedido, el precio unitario, total y el costo del envío, y se cuenta con un área de texto para ingresar algún comentario acerca del pedido, y con cuánto dinero va abonar el mismo.

C.7.1 Reintentar pedido: si el proveedor no toma el pedido en un periodo de 3 minutos, al usuario le aparecerá un pop up diciendo que su pedido no ha podido ser procesado, y le pregunta si desea intentar hacerlo nuevamente.

C.8 Resultado del pedido: Luego de un periodo de tiempo determinado por la demora del pedido, al usuario le aparecerá un pop up donde debe indicar si el pedido fue recibido y donde puede valorar el pedido, con una opción adicional de escribir un comentario.

C.9 Perfil/resumen de cuenta: El usuario accede a una pantalla donde se muestra un resumen de sus pedidos realizados, así como la información sobre los pedidos actuales

Sección cliente proveedor:

P.1: Inicio de sesión de Proveedor: el usuario ingresa a la aplicación y para iniciar sesión debe ingresar un usuario y contraseña como cualquier otro usuario, con la diferencia de que si la cuenta ingresada está asociada a un negocio, se re direccionará a la pantalla principal de gestión del mismo.

P.2 Gestión de productos/promociones:

- **P.2.1 Marcar disponibilidad:** El proveedor dispone de una lista con todos los productos, cada uno con un switch para indicar si está disponible para la venta o no.
- **P.2.2 Modificar precio:** El proveedor dispone de un campo numérico por cada producto donde puede modificar el precio del mismo.
- **P.2.3 Alta de producto:** El proveedor abre un formulario para crear un nuevo producto, donde indica si es un producto regular, una promoción o un gusto de helado(en caso de que el negocio sea una heladería), ingresa una categoría, nombre, descripción y precio.

P.3 Gestionar pedidos: (corresponde a la pantalla que muestra la pila de pedidos)

- **P.3.1 Tomar pedido:** El proveedor acepta un pedido solicitado por un cliente. Este pedido se mueve a una lista de pedidos en proceso, en ese momento al cliente se le notifica que su pedido ha sido aceptado. El pedido caduca en un tiempo si no es atendido o si el cliente lo cancela luego de un cierto tiempo.
    - **P.3.1.1 Marcar pedido como enviado:** El proveedor marca un pedido previamente aceptado, como “en camino”.

  P.3.2 Rechazar pedido: El proveedor puede rechazar un pedido, en ese momento se le notifica al cliente que su pedido ha sido rechazado.

P.5 Solicitar atención: El proveedor indica que necesita la atención del administrador, ya sea para dar de baja productos, actualizar su imagen, problemas técnicos, etc. Para ello tiene un formulario donde ingresa el tipo de solicitud y un comentario sobre la misma.

P.6 Modificar costo de envío: El proveedor accede a un formulario donde puede modificar el costo de los envíos a domicilio.

Sección administrador:

A.1 Inicio de sesión de Administrador: el usuario ingresa a la aplicación y para iniciar sesión debe ingresar un usuario y contraseña.

A.2 Administración de Usuarios

- **A.2.1 Baja Usuario:** El administrador mediante un filtro puede encontrar un determinado usuario y eliminarlo, previo a que se haya recibido una denuncia hacia el mismo.

A.3 Administración Proveedores:

- **A.3.1 ABM Proveedor:**
  - **A.3.1.1 Alta Proveedor:** El administrador puede dar de alta un nuevo proveedor, para esto debe llenar sus datos.
  - **A.3.1.2 Baja Proveedor:** El administrador accede a una lista y selecciona un proveedor para darlo de baja.
  - **A.3.1.3 Modificación Proveedor:** El administrador accede a una lista y selecciona un proveedor para modificar sus datos.

A.4 Administración de negocios:

- **A.4.1 ABM de negocio:**
  - **A.4.1.1 Alta Negocio:** El administrador da de alta un nuevo negocio, lo asocia a un proveedor y llena sus datos
  - **A.4.1.2 Baja Negocio:** El administrador selecciona un negocio de una lista y lo da de baja.
  - **A.4.1.3 Modificación Negocio:** El administrador selecciona un negocio de una lista para modificar sus datos.
- **A.4.2 Ver pedidos de atención:** el administrador accede a una pantalla donde se encuentra una lista a modo de bandeja de entrada con todos pedidos de los proveedores.

A.5 Cobranza: El administrador accede a una pantalla donde se muestran los pagos pendientes de cada negocio, estos pagos tienen la opción de marcarse como pagados. También se pueden modificar los valores de costo de servicio.

ii. Estimaciones

Los parámetros de estimación son los siguientes:

- **Esfuerzo:** valor numérico de 1 a 4, que mide la complejidad de la historia. Siendo 1 una funcionalidad muy simple, por ejemplo una pantalla que solo muestra información, o que tiene muy pocos campos y 4 una funcionalidad de alta complejidad, ya sea por la cantidad de elementos o la interacción de los mismos.
- **Prioridad:** **Baja, Media, Alta**. Indica que historias se deberían realizar primero de acuerdo al valor que tienen para los usuarios.
- **Riesgo:** **Baja, Media, Alta**. Indica la posibilidad de encontrar dificultades, demoras o problemas de implementación al atacar cada historia. Se producen mayores valores de riesgo cuando se trabaja con funcionalidades con atributos no funcionalidades específicos, o que dependen de componentes externos.

Para definir el valor final de cada ítem, cada miembro del equipo asignó un valor dentro de los posibles, justificando su decisión en base a sus conocimientos y experiencia previa, si había discrepancia se consensuaba un valor final, teniendo en cuenta todos los puntos de vista. Los resultados se muestran a continuación:

Cliente final:

Código	Nombre	Esfuerzo	Prioridad	Riesgo en desarrollo
C.1	Inicio de sesión de usuario	1	Media	Baja
C.2	Alta de usuario	1	Media	Media
C.3.1	Ubicación: ingreso manual	2	Alta	Media
C.3.2	Ubicación: ingreso automático	3	Media	Alta
C.4	Elegir rubro	1	Media	Baja
C.5.1	Comercio: elegir por orden alfabético	1	Baja	Baja
C.5.2	Comercio: elegir por cercanía	2	Media	Alta
C.5.3	Comercio: elegir por orden tipo de comidas	1	Baja	Media
C.5.4	Comercio: elegir por popularidad	1	Baja	Media

C.6.1	Pedido: elegir categoría producto	1	**Alta**	**Baja**
C.6.2	Pedido: agregar productos al pedido	2	**Alta**	**Media**
C.7	Confirmar el pedido	2	**Alta**	**Media**
C.7.1	Reintentar pedido	1	**Baja**	**Media**
C.8	Resultado del pedido	2	**Alta**	**Media**
C.9	Perfil/Resumen de cuenta	3	**Baja**	**Media**

Tabla 2-1 Estimaciones cliente final

Cliente proveedor y administrador:

Código	Nombre	Esfuerzo	Prioridad	Riesgo en Desarrollo
P.1	Inicio de sesión de proveedor	1	Media	Baja
P.2	Marcar disponibilidad de productos	2	Media	Media
P.3.1	Tomar pedido	2	Alta	Alta
P.3.1.1	Marcar pedido como enviado	1	Alta	Alta
P.3.2	Rechazar pedido	1	Alta	Alta
P.4	Ver informe de ventas	3	Baja	Baja
P.5	Solicitar atención	1	Baja	Media
A.1	Inicio de sesión de administrador	1	Media	Baja
A.2.1	Baja de usuario	1	Baja	Baja
A.2.2	Informes de uso de usuarios	2	Baja	Baja

A.3.1.1	Alta de proveedor	1	Media	Baja
A.3.1.2	Baja de proveedor	1	Baja	Baja
A.3.1.3	Modificación de proveedor	1	Media	Media
A.4.1.1	Alta de negocio	3	Media	Media
A.4.1.2	Baja de negocio	1	Baja	Baja
A.4.1.3	Modificación de negocio	2	Media	Media
A.4.2	Ver pedidos de atención	1	Baja	Media
A.5	Sistema de cobros	2	Baja	Media

Tabla 2-2 Estimaciones cliente proveedor y administrador

iii. Refinamiento de las historias

Luego de estimar, para cada historia, esfuerzo, prioridad y riesgo de desarrollo, definimos las “tasks” o tareas necesarias para cada una de ellas. Una task consiste en una actividad indivisible, es decir que la realiza una sola persona de principio a fin. Si bien XP define que por cada task se debe detallar en que consiste, al momento de la planificación no contábamos con la información necesaria para hacerlo, por lo que se optó definir de manera genérica 5 tipos de actividades que se realizan durante el desarrollo para definir, por cada historia, cuanto esfuerzo se estimaba tener en cada una de ellas, y así obtener una idea más precisa del esfuerzo total de una historia. La desventaja de esta opción es que no se obtuvo información detallada que guie al desarrollo posterior, y que se tuvo que obtener al momento de desarrollar cada funcionalidad. Por cada historia, se distinguen 5 áreas que delimitan las tareas realizadas. Esta división se realizó con el siguiente criterio:

- **HTML:** corresponde al front-end, incluye todo el desarrollo en html que se necesita para la historia correspondiente.
- **Diseño:** corresponde al front-end, corresponde al trabajo en el diseño estético de la aplicación, ya sea a través de css, imágenes, bootstrap y javascript.
- **JS:** corresponde al desarrollo front-end en javascript, que le brinda la lógica a la aplicación, en nuestro modelo MVC es implementado en controladores y servicios que se encargan de interactuar con el back-end.
- **JAVA:** corresponde al back-end, esta área representa las clases que desarrollaremos para los servicios utilizados, controladores, lógica de negocio y persistencia de datos.
- **BD:** corresponde al back-end, en esta área se incluyen los archivos de generación, ingreso y actualización de las tablas de la base de datos a ser utilizada por el equipo.

Para cada área se estableció una puntuación que se corresponde a la magnitud y/o complejidad de desarrollo que cada historia requiere de ella.

A continuación definimos los task points de las historias de usuarios pertenecientes a los distintos usuarios definidos:

Clientes finales:

Historias		Task points					
ID	**Nombre**	**HTML**	**Diseño**	**JS**	**JAVA**	**BD**	**TOTAL**
C.1	Inicio de sesión de usuario	1	2	1	1	1	**6**
C.2	Alta de usuario	2	2	1	2	1	**8**
C.3.1	Ubicación: ingreso manual	1	1	2	1	1	**6**
C.3.2	Ubicación: ingreso automático	2	3	3	1	1	**10**
C.4	Elegir rubro	1	1	1	1	1	**5**
C.5.1	Comercio: elegir por orden alfabético	1	1	2	1	1	**6**
C.5.2	Comercio: elegir por cercanía	1	1	2	3	1	**8**
C.5.3	Comercio: elegir por orden tipo de comidas	1	1	2	1	1	**6**

C.5.4	Comercio: elegir por popularidad	1	1	2	2	3	**9**
C.6.1	Pedido: elegir categoría producto	2	1	2	1	1	**7**
C.6.2	Pedido: agregar productos al pedido	2	2	2	1	1	**8**
C.7	Confirmar el pedido	2	1	2	1	1	**7**
C.7.1	Reintentar pedido	1	1	1	1	1	**5**
C.8	Resultado del pedido	1	2	1	1	1	**6**
C.9	Perfil/Resumen de cuenta	2	2	2	1	1	**8**

Tabla 2-3 Task points clientes finales

Clientes proveedores y administrador:

Historias		Task points					
ID	**Nombre**	**HTML**	**Diseño**	**JS**	**JAVA**	**BD**	**TOTAL**
P.1	Inicio de sesión de proveedor	1	2	1	1	1	**6**
P.2	Marcar disponibilidad de productos	2	2	1	1	1	**7**
P.3.1	Tomar pedido	1	1	3	2	1	**8**
P.3.1.1	Marcar pedido como enviado	1	1	3	2	1	**8**
P.3.2	Rechazar pedido	1	1	3	2	1	**8**
P.4	Ver informe de ventas	3	2	2	2	1	**10**
P.5	Solicitar atención	2	1	1	1	1	**6**
A.1	Inicio de sesión de administrador	1	1	1	1	1	**5**
A.2.1	Baja de usuario	1	1	1	1	1	**5**

A.2.2	Informes de uso de usuarios	3	1	2	2	1	**9**
A.3.1.1	Alta de proveedor	2	1	1	2	1	**7**
A.3.1.2	Baja de proveedor	1	1	1	1	1	**5**
A.3.1.3	Modificación de proveedor	1	1	1	1	1	**5**
A.4.1.1	Alta de negocio	2	1	2	3	1	**9**
A.4.1.2	Baja de negocio	2	1	2	2	1	**8**
A.4.1.3	Modificación de negocio	2	1	2	3	1	**9**
A.4.2	Ver pedidos de atención	2	1	1	1	1	**6**
A.5	Sistema de cobros	3	1	2	1	1	**8**

Tabla 2-4 Task points clientes proveedores y administrador

iv. Velocidad y tiempo de entrega

Luego del proceso de estimación de las historias de usuario y de las tareas definidas, delimitamos la duración del proyecto y subdividimos el mismo para poder planificar en detalle cada etapa.

A esta unidad de división la llamamos iteración, y está compuesta por un conjunto de historias que realizamos en un periodo limitado de tiempo. El criterio utilizado tiene la finalidad de agrupar historias relacionadas dentro de una misma iteración para que, al final de ésta, se cuente con una funcionalidad completa dentro del sistema.

En la etapa de planificación se definió, a partir de nuestros conocimientos, experiencia y horarios disponibles, una velocidad de equipo de:

- 6 story points por iteración para el periodo que abarca los meses de octubre, noviembre y diciembre (primeras tres iteraciones).
- 7 story points para el periodo correspondiente al año 2016.

La cantidad de story points a la que llegamos es de 51. Al dividir esta cantidad de story points por la velocidad promedio establecida por el equipo (6,5 story point/iteración) nos dio como resultado:

51 story points ÷ 6,5 story point/iteración = 7,84 iteraciones

Aproximando esto resultó en un total de 8 iteraciones para el desarrollo del proyecto.

Consideramos un tiempo razonable de duración por cada iteración de 1 mes.

Consideraciones tenidas en cuenta para el plan de iteraciones:

- Destinamos una semana antes de la primera iteración para realizar una puesta a punto del entorno de desarrollo y para realizar una capacitación inicial sobre las tecnologías, herramientas y lenguajes utilizados.
- Dos semanas de receso durante el mes de enero.
- Consideramos sólo los días de semana para el proyecto, pero por cuestiones externas, como trabajo o estudio, por las cuales no pudimos cumplir con las horas previstas, tomamos tiempo del fin de semana para recuperar dichas horas.

.

Al finalizar el proyecto realizamos un análisis para determinar cuál fue la velocidad real con la que se trabajó, las causas por las cuales se obtuvo ese valor y el impacto que tuvo la misma en el tiempo de entrega:

- **Periodo 1 (octubre 2015 - noviembre 2015):** en el inicio del proyecto consideramos que nuestra velocidad se vio afectada en gran medida por la falta de experiencia con las tecnologías que empleamos y por el hecho de que realizamos una migración a un framework nuevo. **Velocidad: 4.**
- **Periodo 2 (diciembre 2015):** en este periodo nuestra velocidad aumentó por la adquisición de experiencia, pero por otro lado se dispuso de menos tiempo para trabajar en el proyecto, al combinar los hechos de que los dos integrantes nos encontrábamos realizando pasantías y rindiendo las últimas materias de la carrera. **Velocidad: 5.**

- **Periodo 3 (enero 2016 - marzo 2016):** durante este periodo ya se contaba con el conocimiento para avanzar rápidamente sobre las historias, además de aprovechar la reutilización de componentes. Sin embargo uno de nosotros comenzó a trabajar full time por lo que contaba con menos horas para desarrollar. **Velocidad: 7.**
- **Periodo 4 (abril 2016 - agosto 2015):** para la etapa final del proyecto ambos integrantes estábamos trabajando en una jornada de 9 horas por lo que los tiempos se vieron acotados. **Velocidad: 6.**

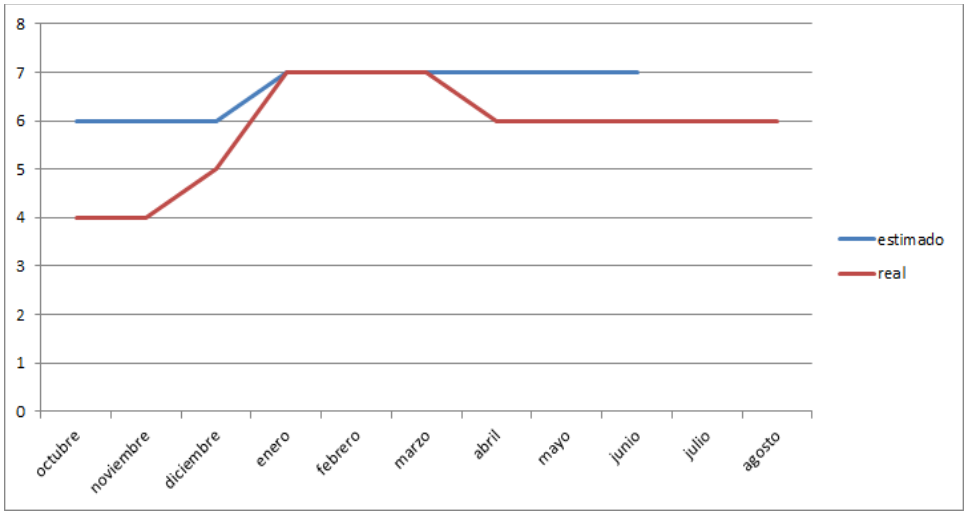

Figura 2-1 Cuadro comparativo de velocidad estimada y real

iv. Diseño

En esta sección se representan un conjunto de diagramas UML, los cuales sirven para comprender el sistema desde distintos aspectos, perspectivas y puntos de vista. Detallamos cada uno de ellos a continuación.

i. Diagramas de clases

El siguiente diagrama permite observar el sistema desde una perspectiva estática, es decir la estructura del mismo y no su comportamiento en ejecución. Se incluyeron las principales entidades que representan al modelo de negocio:

Figura **2-2** Diagrama de clases

ii. Diagrama de actividad

Los siguientes diagramas denominados de actividad pretenden mostrar, de una manera simplificada, la forma en que los distintos procesos interactúan durante la ejecución del sistema:

Armado de pedido:

Muestra el proceso por el cual el cliente utiliza la aplicación para indicar su dirección, elegir un negocio y realizar un pedido con los productos elegidos.

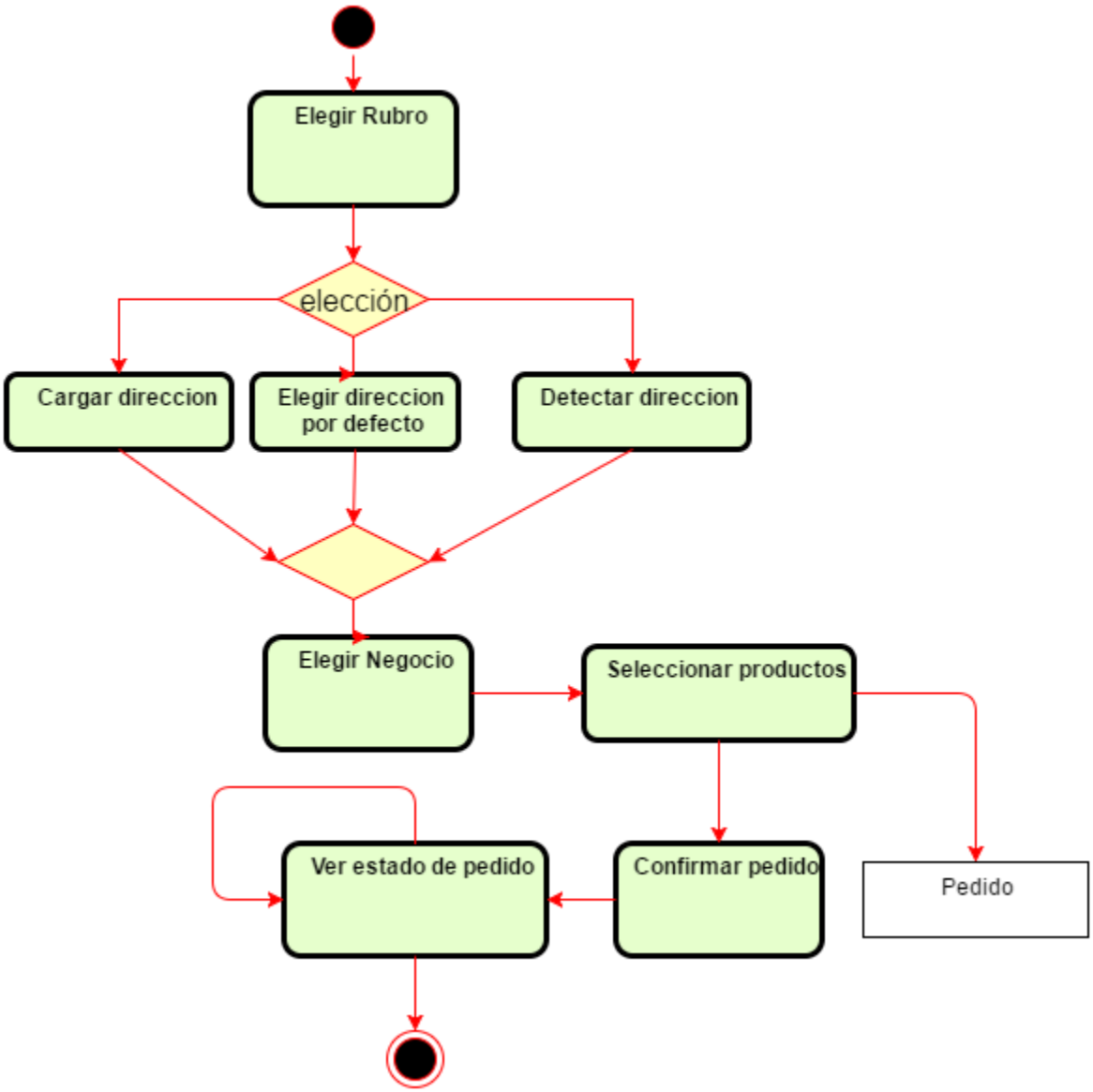

Figura 2-3 Diagrama de actividad - Armado pedido

Gestión de pedidos:

Muestra el proceso por el cual un proveedor gestiona los pedidos que llegan a su negocio.

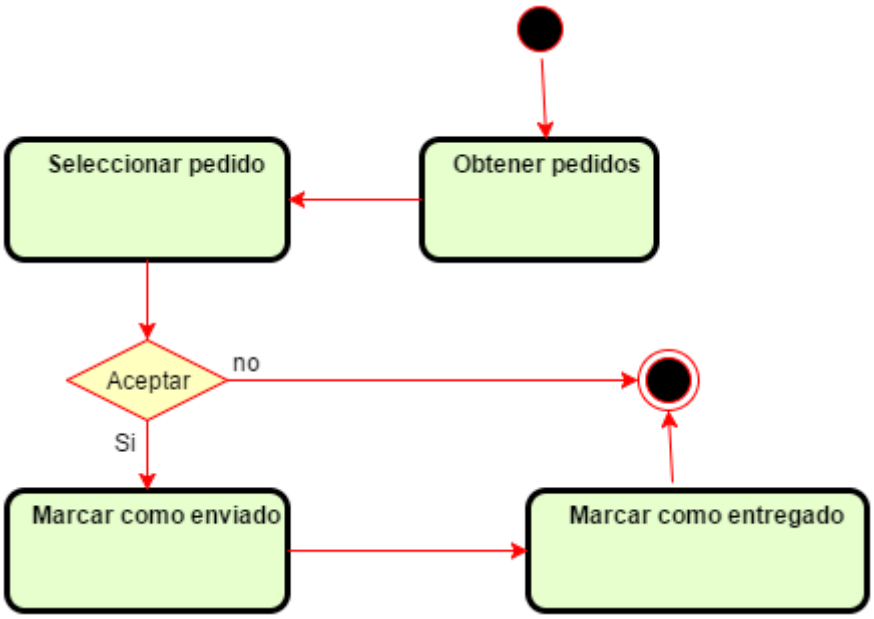

Figura 2-4 Diagrama de actividad - Gestión de los pedidos

Alta, baja y modificación de negocio/proveedor:

Muestra el proceso de creación y mantenimiento de un negocio y de su proveedor asociado.

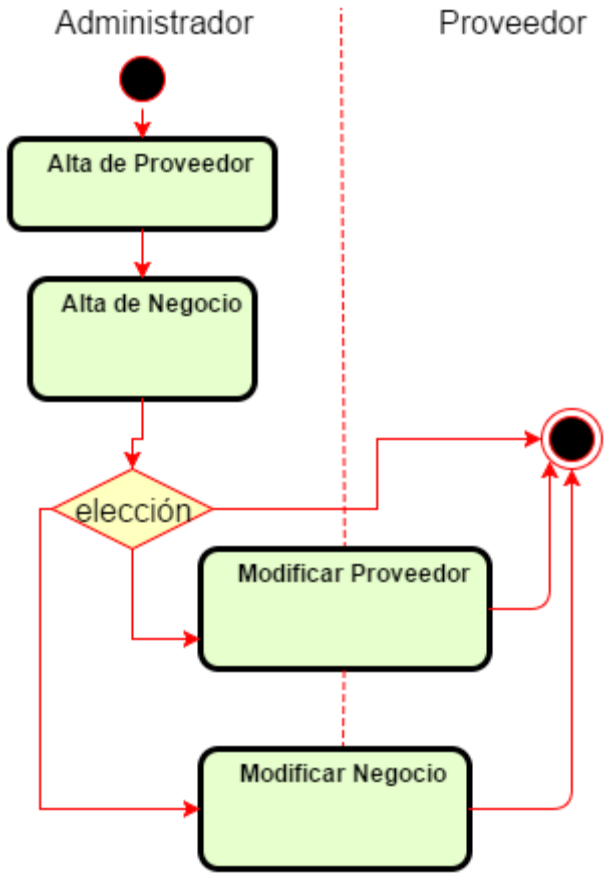

Figura 2-5 Diagrama de actividad - Creación y mantenimiento de negocio

Alta, baja y modificación de cliente:

Muestra el proceso de registro, acciones que puede realizar el usuario.

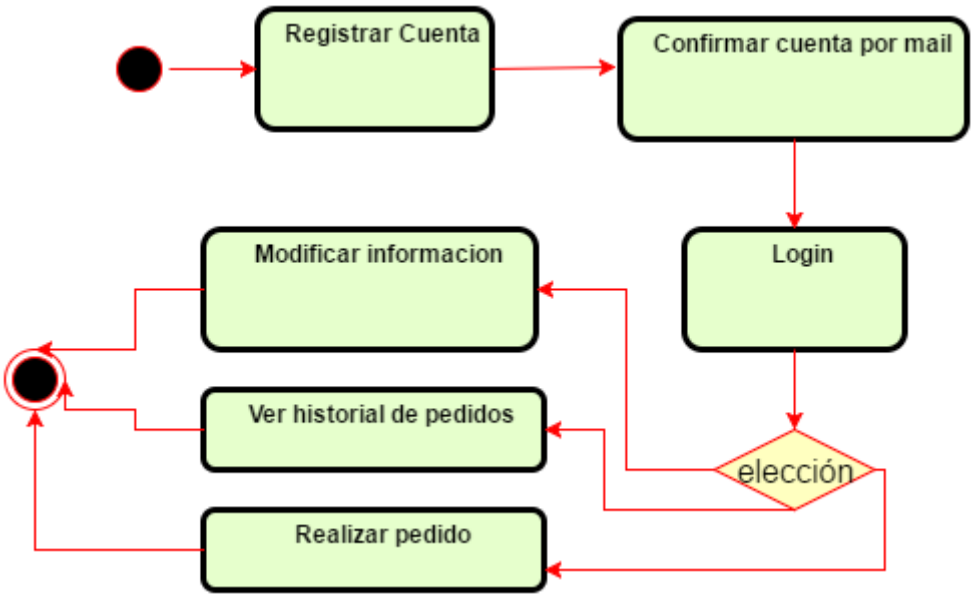

Figura 2-6 Diagrama de actividad - Gestión datos de usuario

b. Etapa iterativa.

i. Implementación.

Tomando como punto de partida el modelo propuesto en la sección anterior, procedemos a describir cómo se implementaron los diseños especificados, con las tecnologías, lenguajes y entornos elegidos.

En las siguientes secciones, describimos las task cards correspondientes a las historias de usuario agrupadas por las iteraciones propuestas.

i. Desarrollo de la Iteración N°1:

Historia: Elegir categoría de producto

Código:	C.6.1	Nombre Hist.:	Elegir categoría de producto	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
12/10/2015	Entidad y repositorio de la entidad Categoría, relación con la entidad Producto.		JAVA, BD	
22/10/2015	Diseño y comienzo con el modelado html y css. Se buscó opciones de listas desplegables en Bootstrap. Se recorren las categorías del Negocio y para cada categoría una nueva sub-lista con los productos pertenecientes a dicha categoría.		HTML, DISEÑO	
22/10/2015	Servicio que trae el Negocio seleccionado.		JS	

Tabla 2-5 Task card historia de usuario C.6.1

Historia: Agregar productos al pedido

Código:	C.6.2	Nombre Hist.:	Agregar productos al pedido	
Desarrollador:	Germán Oñate		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
31/10/2015	Se agregó el servicio que busca productos por negocio. Se generó la entidad Pedido. El servicio está formado por un controlador, la definición del servicio y la implementación del mismo.		JAVA, BD	
2/11/2015	Botón para agregar producto y modal para seleccionar la cantidad. Método en el controlador que maneja el pickup de cantidad.		HTML, JS, DISEÑO	
14/11/2015	Método que crea una entidad producto y asocia la cantidad seleccionada con el producto seleccionado, agregándolo al pedido en construcción.		JS	
17/11/2015	Mejora: agregado de un botón para eliminar líneas individuales de pedido.		HTML, JS, DISEÑO	

Tabla 2-6 Task card historia de usuario C.6.2

Historia: Confirmar el pedido (Resumen de pedido)

Código:	C.7	Nombre Hist.:	Confirmar el pedido (Resumen de pedido)	
Desarrollador:	Germán Oñate		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
13/11/2015	Se agregó un pop-up para confirmar pedido. Con una tabla donde se detalla todos los productos que lo conforman.		HTML, DISEÑO	
14/11/2015	Se terminó el servicio para guardar pedidos, asociado al Negocio por el ID. Este servicio está formado por un controlador.		JAVA, BD	
16/11/2015	Se terminó con la lógica para confirmar pedido, asignándole el correspondiente estado. Queda persistido en la BD.		JS	
16/11/2015	Mejora: se incorporó el precio del envío.		HTML, DISEÑO	

Tabla 2-7 Task card historia de usuario C.7

ii. Desarrollo de la Iteración N°2:

Historia: Tomar pedido

Código:	P.3.1	Nombre Hist.:	Tomar pedido	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
17/11/2015	Tabla que muestra todos los pedidos para un proveedor con un botón que permite tomar pedidos.		HTML	
17/11/2015	Lógica que cambia el estado de un pedido a Aceptado, validando previamente que siga vigente y llama a servicio para persistirlo.		JS	
19/11/2015	Servicio que actualiza un pedido. Este servicio está formado por un controlador.		JAVA, BD	

Tabla 2-8 Task card historia de usuario P.3.1

Historia: Marcar pedido como enviado

Código:	P.3.1.1	Nombre Hist.:	Marcar pedido como enviado	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
23/11/2015	Tabla que muestra todos los pedidos en estado Aceptado con un botón que permite marcar pedidos como enviados. Se respetó el diseño de tabla utilizado en toda la aplicación		HTML, DISEÑO	
24/11/2015	Lógica que cambia el estado de un pedido a “En camino” y reutilización del servicio para persistirlo.		JS	

Tabla 2-9 Task card historia de usuario P.3.1.1

Historia: Rechazar pedido

Código:	P.3.2	Nombre Hist.:	Rechazar pedido	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
26/11/2015	Tabla que muestra todos los pedidos con un botón que permite rechazar. Se respetó el diseño de tabla utilizado en toda la aplicación.		HTML, DISEÑO	
27/11/2015	Lógica que cambia el estado de un pedido a “Rechazado” y llama a servicio para persistirlo, para ello reutilizamos el servicio para actualizar un pedido.		JS	

Tabla 2-10 Task card historia de usuario P.3.2

Historia: Elegir rubro

Código:	C.4	Nombre Hist.:	Elegir rubro	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
1/12/2015	Se definieron los rubros iniciales con los que vamos a trabajar y se los asoció a la entidad Negocio.		JAVA	
4/12/2015	Se creó pantalla con botones que redirigen a sus respectivos rubros.		HTML	
4/12/2015	Lógica que llama a un servicio que recupera todos los negocios con un determinado rubro.		JS	
9/12/2015	Servicio que recupera entidades Negocios con el criterio de búsqueda “Rubro”.		JAVA	
11/12/2015	Mejoras: se realizaron varios rediseños de esta sección ya que es la presentación inicial de la aplicación.		HTML, DISEÑO	

Tabla 2-11 Task card historia de usuario C.4

iii. Desarrollo de la Iteración N°3:

Historia: Ingreso manual

Código:	C.3.1	Nombre Hist.:	Ingreso manual	
Desarrollador:	Germán Oñate		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
14/12/2015	Se creó modal que permite el ingreso de la dirección. Campos: dirección, departamento, piso y select para la ciudad.		HTML, DISEÑO	
16/12/2015	Lógica que asocia los datos referidos a la dirección del usuario con el pedido.		JS	

Tabla 2-12 Task card historia de usuario C.3.1

Historia: Resultado pedido

Código:	C.8	Nombre Hist.:	Resultado pedido	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
18/01/2016	Se implementó un servicio que permite persistir el pedido creado y un servicio para obtener el estado del pedido actual.		JAVA, BD	
20/01/2016	Se implementó la lógica del lado del cliente para obtener periódicamente el estado del pedido y realizar el aviso correspondiente en caso de que haya cambiado.		JS	
27/01/2016	Se utilizó la barra navegación para identificar por colores los distintos estados del pedido y se implementaron los mensajes con el detalle correspondiente a cada estado.		HTML, DISEÑO, JS	

Tabla 2-13 Task card historia de usuario C.8

Historia: Marcar disponibilidad de productos

Código:	P.2	Nombre Hist.:	Marcar disponibilidad de productos	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
25/12/2015	Se implementó un servicio para obtener los productos de un proveedor y otro para actualizarlos.		JAVA, BD	
18/01/2016	Se implementó una pantalla donde se listan los productos agrupados por categoría, cada uno con un switch para cambiar su estado.		HTML, DISEÑO	
25/01/2016	Se modificó la lista de productos que visualiza el cliente para que no le permita elegir los productos no disponibles.		HTML, DISEÑO	
2/03/2016	Mejoras: se incorporó la modificación del precio de cada producto dentro de la misma pantalla.		HTML, JS	
9/03/2016	Mejoras: se incorporaron promociones como otro tipo de producto y se agregó un alta para los productos, funcionalidad que sólo estaba planteada para el administrador.		HTML, JS, JAVA, BD	

Tabla 2-14 Task card historia de usuario P.2

iv. Desarrollo de la Iteración N°4:

Historia: Ingreso automático

Código:	C.3.2	Nombre Hist.:	Ingreso automático	
Desarrollador:	Germán Oñate		Story Points:	3
Fecha	Realizado		Áreas	Comentarios
9/02/2016	En el modal ya creado para el ingreso manual se agregó un botón para detectar automáticamente la dirección del usuario y guardar sus coordenadas.		HTML	
9/02/2016	Se agregó librería de Google para georreferenciar posición del usuario, se incluyó en las dependencias del proyecto (bower.json) y se configuró el mismo con la api key obtenida.		JS	
11/02/016	Se terminó la lógica del controlador que asigna la dirección del usuario al pedido.		JS	

Tabla 2-15 Task card historia de usuario C.3.2

Historia: Filtrar búsqueda por orden alfabético

Código:	C.5.1	Nombre Hist.:	Filtrar búsqueda por orden alfabético	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
4/03/2016	Se agregó un botón en el navbar de negocios.html.		HTML	
7/03/2016	Se agregó un filtro por orden alfabético que trabaja sobre la lista de negocios recuperados de la base. Es por orden ascendente y descendente. Para recuperar la lista de Negocios se creó un servicio en el backend.		JS, JAVA	

Tabla 2-16 Task card historia de usuario C.5.1

Historia: Filtrar búsqueda por cercanía

Código:	C.5.2	Nombre Hist.:	Filtrar búsqueda por cercanía	
Desarrollador:	Germán Oñate		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
5/03/2016	Se desarrolló un algoritmo en el controlador que recorre los negocios y calcula la distancia entre las coordenadas del negocio y las del usuario.		JS	
5/03/2016	Se agregó un botón en el navbar de negocios.html.		HTML	
7/03/2016	Se agregó un filtro por cercanía de cada negocio a la tabla de los mismos.		JS	

Tabla 2-17 Task card historia de usuario C.5.2

Historia: Reintentar pedido

Código:	C.7.1	Nombre Hist.:	Reintentar pedido	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
20/03/2016	Se implementó la lógica del lado del cliente para que luego de una cierta cantidad de intentos de ver el estado del pedido sin cambios, le pida confirmación al usuario para seguir intentando o cancelar el pedido.		JS	
2/04/2016	Se implementó el modal de confirmación para que el usuario ingrese si quiere cancelar o no el pedido.		HTML	

Tabla 2-18 Task card historia de usuario C.7.1

v. Desarrollo de la Iteración N°5:

Historia: Inicio de sesión de usuario

Código:	C.1	Nombre Hist.:	Inicio de sesión de usuario	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
4/04/2016	Se adaptó el html propuesto por el framework con el diseñado (ubicación).		HTML, DISEÑO	
6/04/2016	Se completó el servicio de inicio de sesión para redirigir en base al rol del usuario al perfil al que pertenece. Para el rol usuario se redirige al state homeUser.		JS	
7/04/2016	Se modificó el servicio en el backend. Se agregaron parámetros en el response.		JAVA	
11/04/2016	En el home del Usuario, en base al usuario que inició sesión se llama a servicio para traer los datos del Usuario dado de alta.		JS	

Tabla 2-19 Task card historia de usuario C.1

Historia: Alta usuario

Código:	C.2	Nombre Hist.:	Alta usuario	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
5/04/2016	Se creó entidad Usuario y su relación con la entidad User que es nativa del framework. Se crearon servicios para recuperar usuario por distintos criterios.		JAVA, BD	
6/04/2016	Se creó formulario con los campos definidos para la entidad Usuario, sumados a la de la entidad User.		HTML	
7/04/2016	Servicio en el controlador que valida los campos y luego llama al servicio para guardar el cliente.		JS	

Tabla 2-20 Task card historia de usuario C.2

Historia: Inicio de sesión de proveedor

Código:	P.1	Nombre Hist.:	Inicio de sesión de proveedor	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
11/04/2016	Se agregó validación al controlador de inicio de sesión, para que cuando el rol sea de proveedor redirige al state gestionPedidos. Se reutiliza el html de inicio de sesión para todos los roles		JS, HTML	
13/04/2016	Se modificó el servicio en el backend. Se agregaron parámetros en el response		JAVA	
20/04/2016	En el home del Proveedor, en base al usuario que inició sesión se llama a servicio para traer los datos del Proveedor dado de alta		JS	

Tabla 2-21 Task card historia de usuario P.1

Historia: Alta proveedor

Código:	A.3.1.1	Nombre Hist.:	Alta proveedor	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
12/04/2016	Se creó entidad Proveedor y su relación con la entidad User. Se crearon servicios para recuperar proveedor por distintos criterios		JAVA, BD	
5/05/2016	Se creó formulario d con los campos definidos para la entidad Proveedor, sumados a la de la entidad User dentro del perfil Administrador		HTML	
22/04/2016	Servicio en el controlador que valida los campos y luego llama al servicio para guardar el proveedor		JS	

Tabla 2-22 Task card historia de usuario A.3.1.1

Historia: Alta negocio

Código:	A.4.1.1	Nombre Hist.:	Alta negocio	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
12/04/2016	Se creó entidad Negocio y su relación con la entidad Proveedor. Se crearon servicios para recuperar Negocio por distintos criterios.		JAVA, BD	
5/05/2016	Se añadió al formulario de alta proveedor los campos pertenecientes a la entidad Negocio.		HTML	
25/04/2016	Se agregaron las validaciones correspondientes a la entidad Negocio, y dentro del método save la llamada al servicio para dar de alta el negocio.		JS	

Tabla 2-23 Task card historia de usuario A.4.1.1

vi. Desarrollo de la Iteración N°6:

Historia: Modificación negocio

Código:	A.4.1.3	Nombre Hist.:	Modificación negocio	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
14/05/2016	Servicio en el backend para actualizar datos de un negocio.		JAVA,BD	
16/05/2016	En el perfil Administrador se agregó el ABM de la entidad Negocio, donde se modeló un formulario con todos los datos pertenecientes a la entidad.		HTML	
19/05/2016	Se agregaron las validaciones correspondientes en todos los campos del formulario. Método que se comunica con el servicio del backend.		JS, HTML	

Tabla 2-24 Task card historia de usuario A.4.1.3

Historia: Filtrar búsqueda por tipo de comida

Código:	C.5.3	Nombre Hist.:	Filtrar búsqueda por tipo de comida	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
16/05/2016	Se agregó un select en el navbar de negocios.html con un listado de categorías.		HS	
16/05/2016	En el controlador se desarrolló un método que filtra la lista de negocios en base a la categoría seleccionada. Recorre los negocios consultando las categorías de los mismos.		JS	

Tabla 2-25 Task card historia de usuario C.5.3

Historia: Filtrar búsqueda por popularidad

Código:	C.5.4	Nombre Hist.:	Filtrar búsqueda por popularidad	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
21/03/2016	Se desarrolló algoritmo en el controlador que recorre los negocios y saca un porcentaje de popularidad basado en a las opiniones positivas y negativas.		JS	
18/05/2016	Se agregó botón en el navbar de negocios.html.		HTML	
18/05/2016	Se agregó filtro por popularidad de cada negocio a la tabla de los mismos.		JAVA, BD	

Tabla 2-26 Task card historia de usuario C.5.4

Historia: Cobranza

Código:	A.5	Nombre Hist.:	Cobranza	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
30/05/2016	Se realizó un servicio que busca todos los pedidos por cada negocio registrado y a partir de un parámetro de cobranza se calcula lo que deben abonar por el servicio para el último mes.		JAVA, BD	
5/06/2016	Se realizó un buscador para obtener la información de cobranza para un mes en particular y permite marcar como cobrada su deuda. Además informa el total.		HTML, JS JAVA	

Tabla 2-27 Task card historia de usuario A.5

Historia: Inicio de sesión de administrador

Código:	A.1	Nombre Hist.:	Inicio de sesión de administrador	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
9/05/2016	Se agregó validación al controlador de inicio de sesión, para que cuando el rol sea de proveedor redirige al state homeAdmin. Se reutiliza el html de inicio de sesión para todos los roles.		JS, HTML	
10/05/2016	Se modificó el servicio en el backend. Se agregaron parámetros en el response		JAVA	
10/05/2016	En el home del Administrador, en base al usuario que inició sesión se llama a servicio para traer los datos del Administrador dado de alta.		JS	

Tabla 2-28 Task card historia de usuario A.1

vii. Desarrollo de la Iteración N°7:

Historia: Perfil/resumen de cuenta

Código:	C.9	Nombre Hist.:	Perfil/resumen de cuenta	
Desarrollador:	Santiago Guichard		Story Points:	3
Fecha	Realizado		Áreas	Comentarios
8/05/2016	Se implementó el servicio para obtener los pedidos que no están finalizados para la sección de pedidos actuales, y otro para obtener los pedidos finalizados para mostrar en la sección “historial” de pedido.		JAVA, BD	
15/05/2016	Se implementó la lógica del lado del cliente para realizar un nuevo pedido con los datos de un pedido anterior.		JS	
18/05/2016	se implementó la vista para mostrar ambas tablas de pedidos, y los formularios de confirmación, comentar pedido y actualización de datos.		HTML, DISEÑO, JS	
20/05/2016	se rediseño la pantalla para mostrar de forma más atractiva es estado de cada pedido y ordenar mejor la información.		HTML, DISEÑO, JS	

Tabla 2-29 Task card historia de usuario C.9

Historia: Ver informe de ventas

Código:	P.4	Nombre Hist.:	Ver informe de ventas	
Desarrollador:	Germán Oñate		Story Points:	3
Fecha	Realizado		Áreas	Comentarios
29/04/2016	Se investigó sobre librerías para trabajar con gráficos estadísticos.			
	CANCELADA.			

Tabla 2-30 Task card historia de usuario P.4

Historia: Solicitar atención

Código:	P.5	Nombre Hist.:	Solicitar atención	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
11/04/2016	Se implementó el servicio para guardar un mensaje de solicitud de atención asociado al negocio que lo pide.		JAVA, JS, BD	
16/04/2016	se realizó el formulario donde se elige la categoría y se carga el mensaje y se agregó la opción al menú de herramientas del negocio.		HTML, DISEÑO, JS	

Tabla 2-31 Task card historia de usuario P.5

viii. Desarrollo de la Iteración N°8:

Historia: Baja usuario

Código:	A.2.1	Nombre Hist.:	Baja usuario	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
3/06/2016	Servicio en el backend que da de baja una entidad Usuario, que tiene como parámetro el ID del mismo.		JAVA, BD	
13/06/2016	En el perfil Administrador, se agregó en la sección ABM de Usuario un botón para darlo de baja. Este botón llama a un método que se comunica con el servicio del backend.		HTML, JS	

Tabla 2-32 Task card historia de usuario A.2.1

Historia: Informes de uso de usuarios

Código:	A.2.2	Nombre Hist.:	Informes de uso de usuarios	
Desarrollador:	Santiago Guichard		Story Points:	2
Fecha	Realizado		Áreas	Comentarios
19/05/2016	CANCELADA			

Tabla 2-33 Task card historia de usuario A.2.2

Historia: Baja proveedor

Código:	A.3.1.2	Nombre Hist.:	Baja proveedor	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
27/06/2016	Servicio en el backend que da de baja una entidad Proveedor, que tiene como parámetro el ID del mismo.		JAVA, BD	
30/06/2016	En el perfil Administrador, se agregó en la sección ABM de Proveedores un botón para darlo de baja. Este botón llama a un método que se comunica con el servicio del backend.		HTML, JS	

Tabla 2-34 Task card historia de usuario A.3.1.2

Historia: Modificación proveedor

Código:	A.3.1.3	Nombre Hist.:	Modificación proveedor	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
27/06/2016	Servicio en el backend para actualizar datos de un proveedor.		JAVA,BD	
29/06/2016	En el perfil Administrador se agregó el ABM de la entidad Proveedor, donde se modeló un formulario con todos los datos pertenecientes a la entidad.		HTML	
6/07/2016	Se agregaron las validaciones correspondientes en todos los campos del formulario. Método que se comunica con el servicio del backend.		JS, HTML	

Tabla 2-35 Task card historia de usuario A.3.1.3

Historia: Baja negocio

Código:	A.4.1.2	Nombre Hist.:	Baja negocio	
Desarrollador:	Germán Oñate		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
4/06/2016	Servicio en el backend que da de baja una entidad Negocio, que tiene como parámetro el ID del mismo.		JAVA, BD	
11/06/2016	En el perfil Administrador, se agregó en la sección ABM de Negocios un botón para darlo de baja. Este botón llama a un método que se comunica con el servicio del backend.		HTML, JS	

Tabla 2-36 Task card historia de usuario A.4.1.2

Historia: Ver pedidos de atención.

Código:	A.4.2	Nombre Hist.:	Ver pedidos de atención	
Desarrollador:	Santiago Guichard		Story Points:	1
Fecha	Realizado		Áreas	Comentarios
16/03/2016	Se implementó el servicio para obtener todos los pedidos de atención.		JAVA , BD	
1/05/2016	Se implementó la vista donde se listan los pedidos con sus detalles y se implementó la función para cambiar su estado.		HTML, DISEÑO, JS	

Tabla 2-37 Task card historia de usuario A.4.2

ii. Pruebas funcionales

Formulación del plan de pruebas:

Para determinar el plan de pruebas a seguir tomamos como referencia las historias de usuario y la tabla de descripción de validaciones requeridas para las entradas del usuario. En función de estos datos se elaboró una planilla de prueba que cuenta con los siguientes campos:

- Número de caso: identificador numérico único para cada caso.
- Historia: código de historia que hace referencia a la funcionalidad que pretende probar el caso
- Caso de prueba: descripción de lo que se va a probar en el caso
- Resultado esperado: descripción de cómo debería responder el sistema basándose en la historia de usuario
- Resultado obtenido: aprobado si el resultado coincide con lo esperado, no aprobado en caso contrario.

Para los casos no aprobados se dispone de otra tabla con los siguientes campos:

- Número de caso.
- Comentario: explicación de cómo reproducir el error o porque se dio el caso como no aprobado.
- Solución: pasos seguidos para la corrección del error.

Ejecución de las pruebas: la planilla con el resultado de las pruebas se puede encontrar en el anexo.

iii. Contingencias

Tener formalizado un plan para gestionar el riesgo nos brindó la posibilidad de tener de antemano una forma de actuar ante las distintas contingencias que surgen durante el desarrollo de un proyecto de este tamaño. La decisión de basarnos en un modelo taxonómico nos permitió identificar de manera rápida los distintos ámbitos que implican el desarrollo del proyecto, y sus riesgos asociados, conocimiento que sería imposible de obtener a partir de la experiencia con la que contamos.

Las contingencias a las cuales nos tuvimos que enfrentar durante el desarrollo fueron las siguientes.

- **Modificación y aparición de nuevos requerimientos:** al ir avanzando sobre distintas funcionalidades, y evaluando su forma de resolución se fueron detectando nuevos requerimientos, por lo que, siguiendo el plan de contingencia, se procedió a evaluar su prioridad y dificultad relativa a los requerimientos actuales para tomar una decisión acerca de cómo continuar.
- **Reemplazo de tecnologías utilizadas:** En las etapas tempranas del desarrollo del sistema, nos encontramos con la posibilidad de modificar completamente la estructura del proyecto, con el fin de utilizar un framework que nos permitiría avanzar más rápido a largo plazo. Por lo que se decidió suspender el desarrollo de nuevas funcionalidades para dedicarse de lleno a la migración del proyecto, lo que inicialmente impactó gravemente en los tiempos del proyecto, pero que gradualmente se fue corrigiendo sobre el final del desarrollo, al poder aprovechar las ventajas que el framework facilitaba.

c. Etapa final

i. Pruebas de performance

Realizamos pruebas de performance/rendimiento para determinar el costo que tiene, para el sistema, realizar un conjunto pautado de operaciones en determinadas condiciones de trabajo. La idea es validar la calidad del mismo y evaluar atributos como escalabilidad, fiabilidad y utilización de los recursos. Además, intentamos detectar “cuellos de botella” para que nuestros usuarios no sufran caídas del servicio.

La explicación de la herramienta y el resultado de las pruebas se adjuntan en el anexo.

ii. Puesta en producción

Al finalizar cada iteración, en la etapa que denominamos “despliegue”, montamos servidores locales en los cuales testeamos las funcionalidades definidas en las historias pertenecientes a la iteración. Se puede seguir con más detalle en la sección “Manual de Instalación” incluida en el anexo.

Por un tema de costo, una vez finalizada la última iteración alquilamos un servidor *amazon*[3] donde se encuentra en ejecución la aplicación en modo de prueba. Brindamos información detallada de este servicio y su configuración en la sección “Tecnologías y entorno de desarrollo”.

[3] Colección de servicios de computación en la nube - https://aws.amazon.com/es/

3. GALERIA DE IMÁGENES

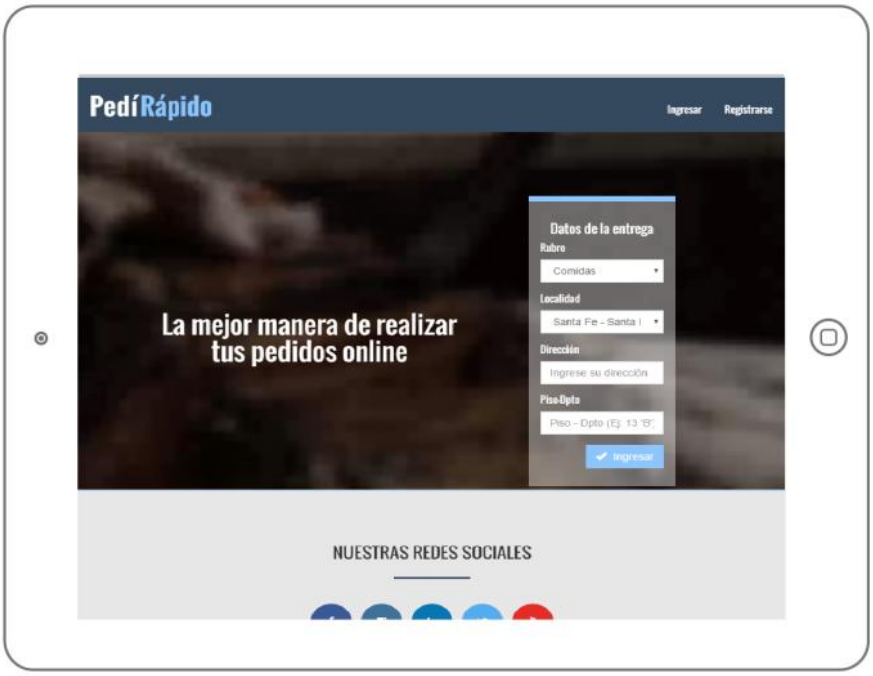

Figura 3-1 Página de inicio

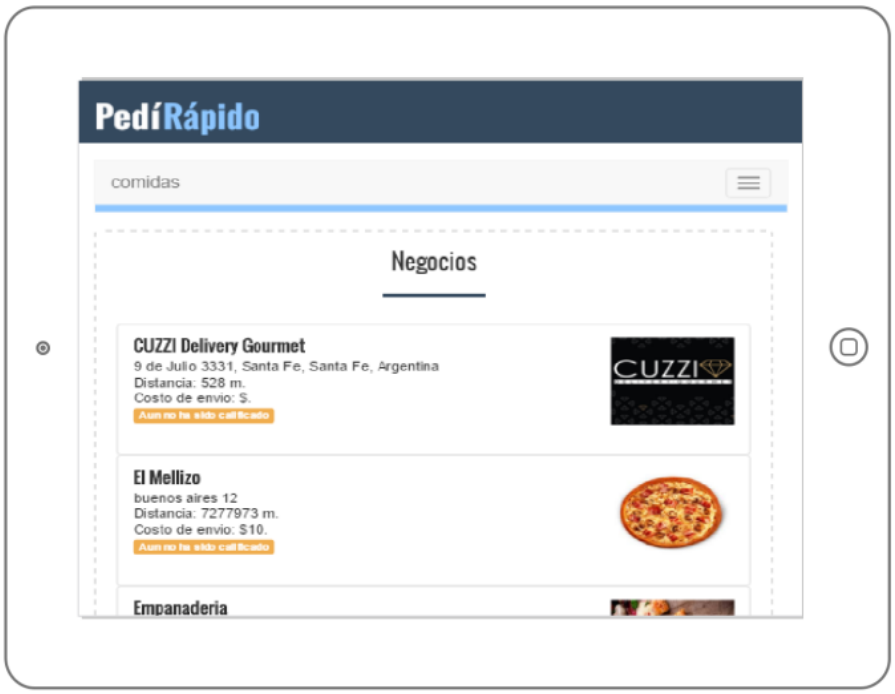

Figura 3-2 Usuario - Listado de negocios

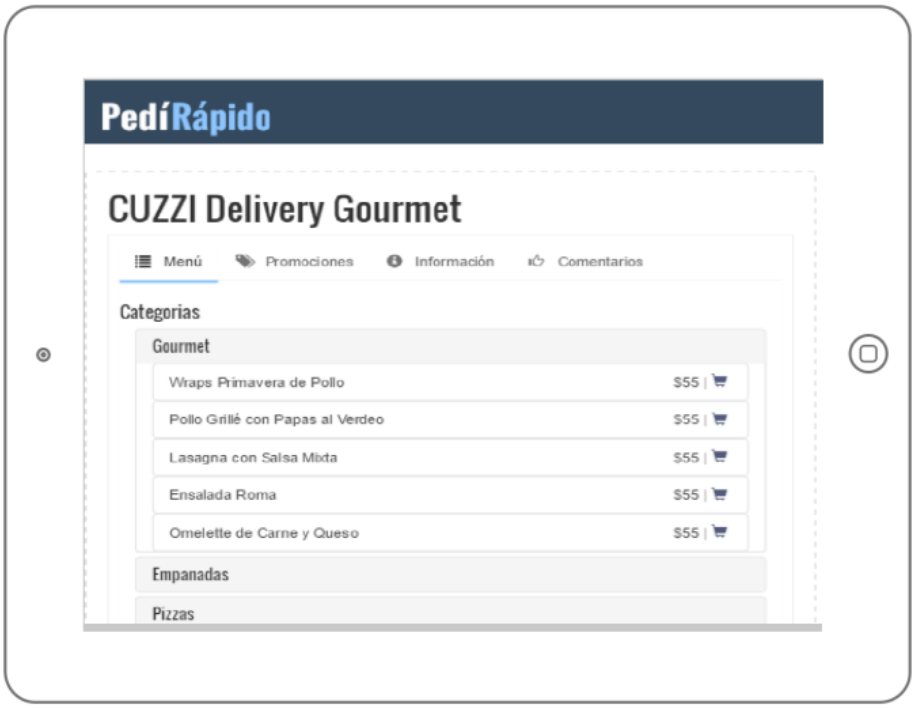

Figura 3-3 Usuario - Productos de negocio elegido

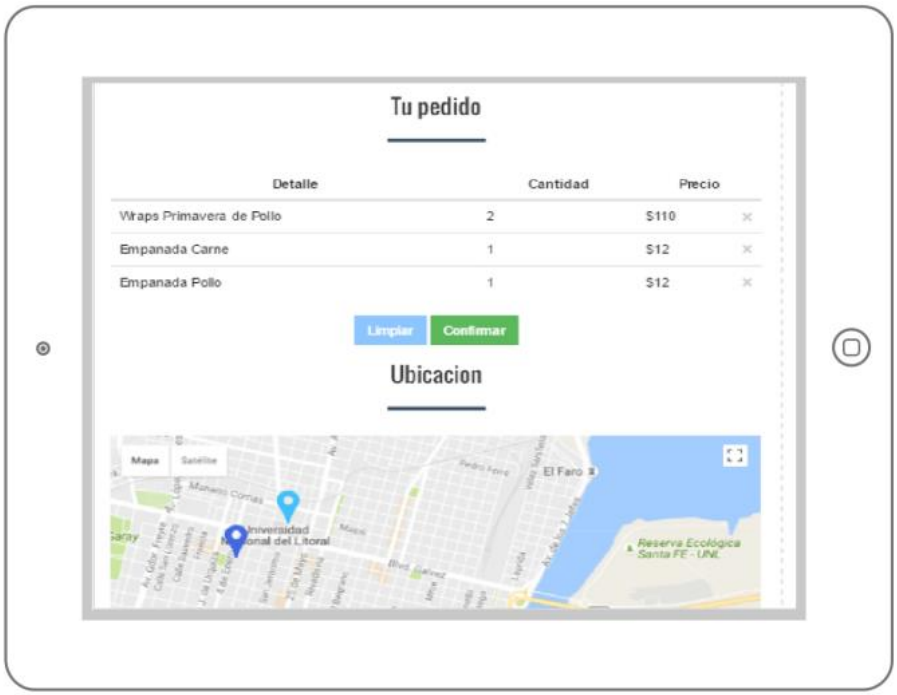

Figura 3-4 Usuario - Pedido ordenado y ubicación del negocio

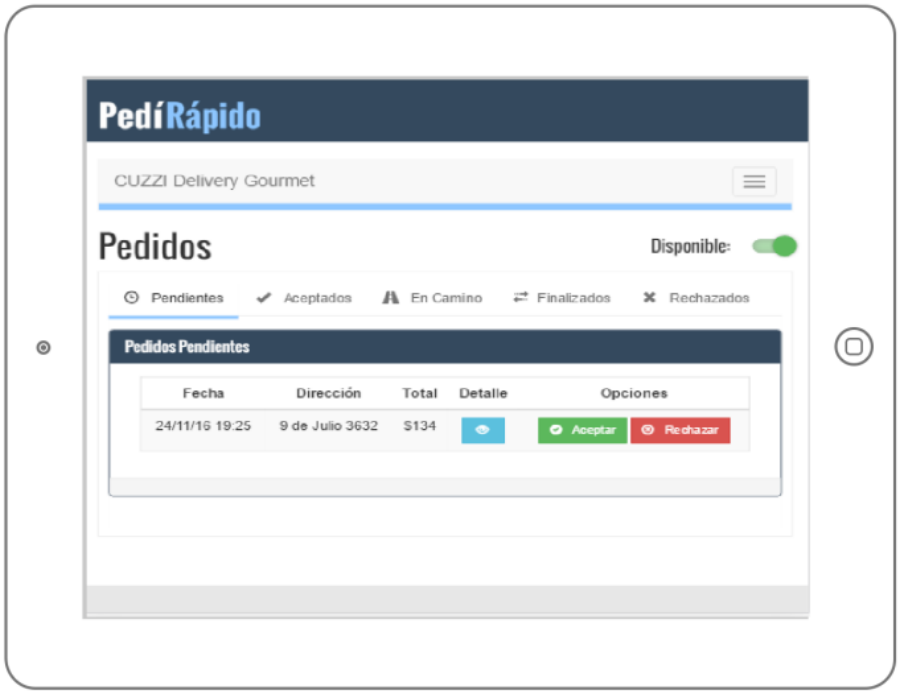

Figura 3-5 Negocio - Tabla de pedidos

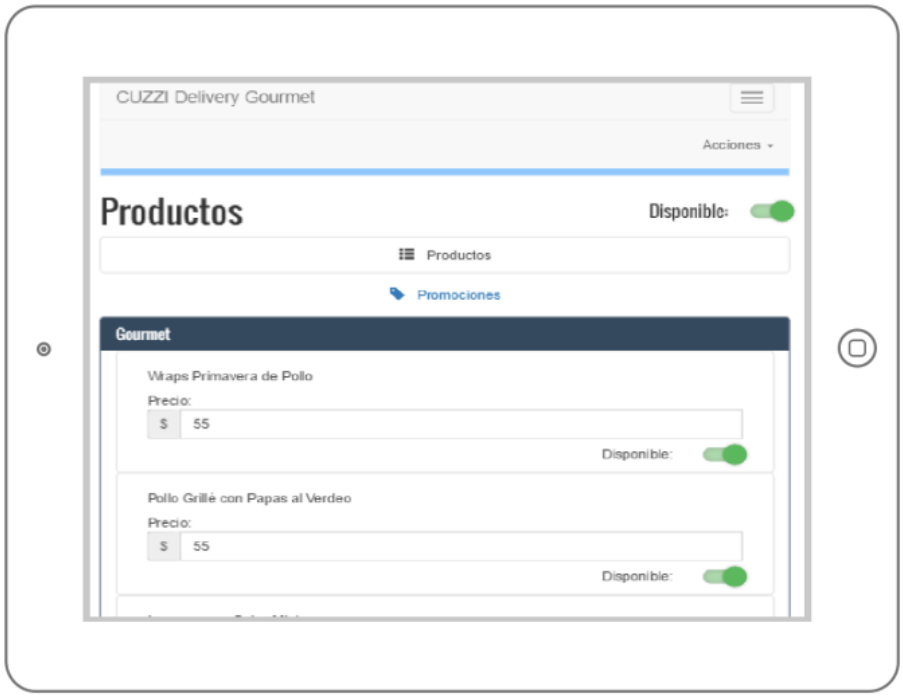

Figura 3-6 Negocio - Modificación de productos

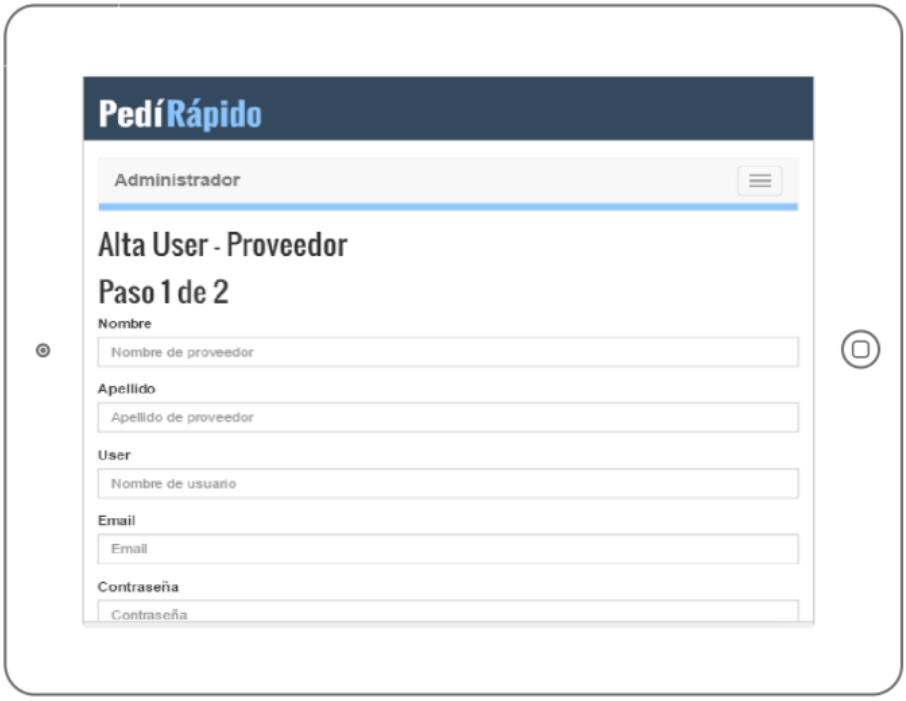

Figura 3-7 Administrador - Alta de negocio

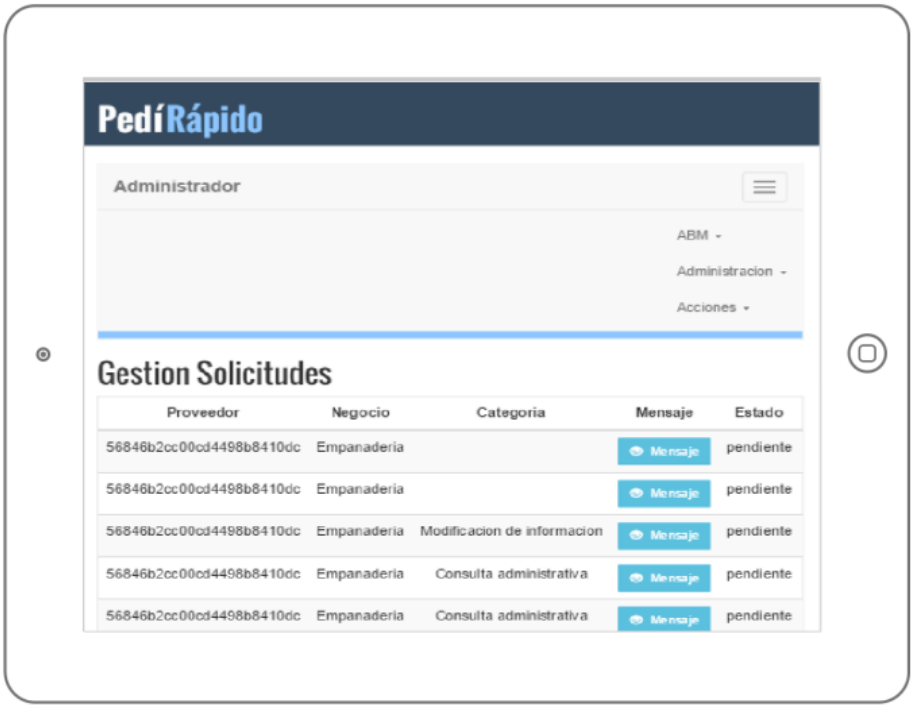

Figura 3-8 Administrador - Gestión de solicitudes

4. ARQUITECTURA DEFINIDA Y REQUERIMIENTOS NO FUNCIONALES

A continuación describiremos la arquitectura utilizada y los requerimientos no funcionales tenidos en cuenta para el desarrollo del proyecto.

Documento de la arquitectura inicial:

Para la documentación de la arquitectura inicial aplicamos el modelo de arquitectura estudiado en la facultad “4+1”. Es un modelo de vistas que encaja con el estándar “IEEE 1471-2000” (Recommended Practice for Architecture Description of Software-Intensive Systems) que se utiliza para describir la arquitectura de un sistema software basado en el uso de múltiples puntos de vista.

Vista física:

En esta vista se muestra desde la perspectiva de un ingeniero de sistemas todos los componentes físicos del sistema, así como las conexiones físicas entre esos componentes que conforman la solución (incluyendo los servicios).

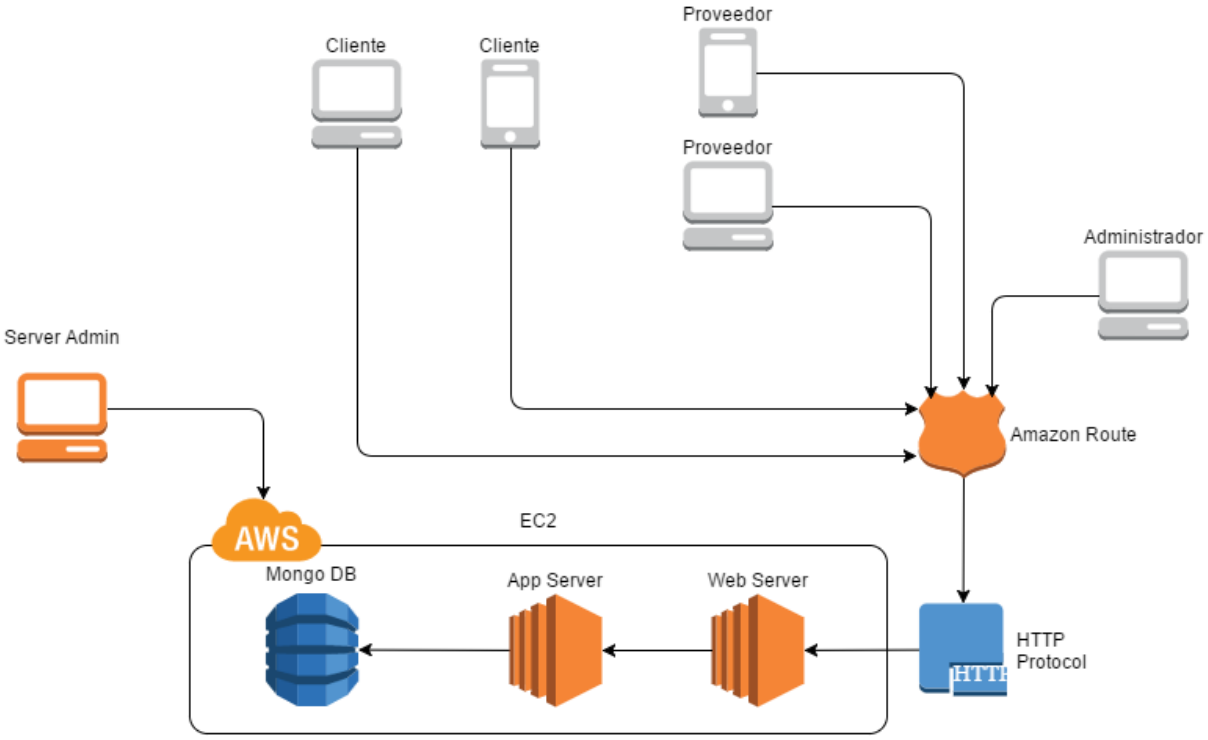

Figura 4-1 Vista física del sistema

La elección de esta arquitectura se realizó a fin de satisfacer los siguientes atributos no funcionales:

- **Escalabilidad:** Si bien el sistema se planteó para un escenario regional, es deseable que éste se pueda ir adaptando a cambios crecientes en su uso sin requerir mayores modificaciones. Por ejemplo, se podrían agregar réplicas o redundancia a alguno de los servidores.

Aplicando escalabilidad con AWS:

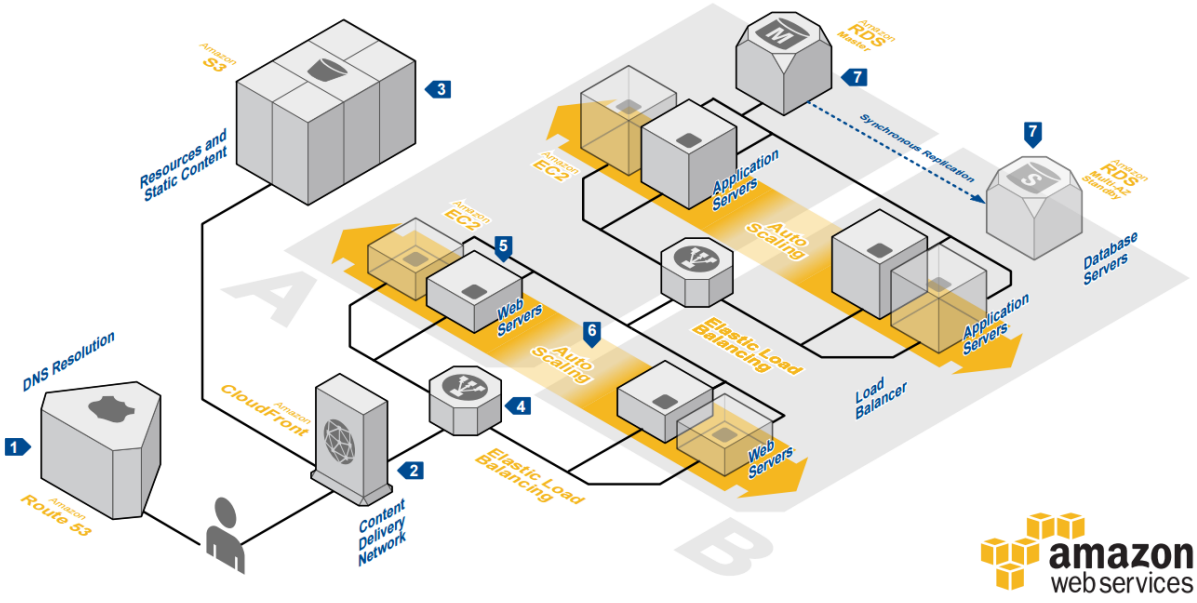

Figura 4-2 Arquitectura servidores amazon

1. **Amazon Route:** DNS de alta disponibilidad que dirige el tráfico a la infraestructura de AWS
2. **Amazon CoudFront:** Servicio que re direcciona solicitudes de contenido estático, dinámico y streaming al Storage System más cercano
3. **Storage System:** Almacén de contenido estático, dinámico y streaming.

4. **Elastic Load Balancer:** distribuye el tráfico entrante entre las múltiples instancias EC2.
5. **Web Server:** Servidor web desplegado en una instancia EC2 configurable según las necesidades
6. **Auto Scaling group:** administra automáticamente la cantidad de instancias para soportar el crecimiento y las variaciones de tráfico.

● **Modificabilidad:** Esta arquitectura dividida en distintas bandas permite delimitar distintos aspectos del problema del cambio y atacarlos individualmente. El hecho de optar por una aplicación web permite que cualquier cambio en la misma se realice transparentemente para los usuarios, que no necesitan preocuparse por instalar actualizaciones, y posibilita una fácil inserción de una aplicación móvil, que se encuentra entre las próximas expectativas para este proyecto. En nuestra solución, las bandas se encuentran delimitadas por interfaces bien definidas que junto al acompañamiento de estándares y tecnologías permiten que un cambio no repercuta transversalmente en la aplicación. Por ejemplo, al utilizar web API REST para comunicar el front-end con el back-end, se podría modificar o agregar distintas implementaciones de servicios incluso con distintas tecnologías (php, .NET) sin tener que realizar cambios en el front-end.

● **Seguridad:** debido al manejo de información sensible y crítica, y que a la aplicación pueden acceder distintas personas con distintos perfiles e información asociada a los mismos (todos ellos desde sitios no confiables), creemos muy importante controlar estos aspectos, ya sea implementando soluciones de seguridad en profundidad a lo largo de las distintas capas de la aplicación y reduciendo la superficie de ataque de los mismas.

Para lograr esto se utilizaron diversas tecnologías y estándares que se nombran a continuación:

Oauth2: es la evolución del protocolo Oauth, permite flujos simples de autorización para sitios web o aplicaciones web. Proporciona a los usuarios un acceso a sus datos al mismo tiempo que protege las credenciales de su cuenta. Este protocolo está implementado en el módulo **Spring Security**, que además de implementar las acciones de autenticación y autorización, permite la protección contra ataques como session fixation[4], clickjacking[5], cross site request forgery[6], etc.

Logback: Framework de logeo de operaciones implementado por JHipster para registrar todas las operaciones que ocurren en la aplicación (por ejemplo llamadas rest, y comandos de base de datos).

MongoDB Encrypted Storage Engine: Permite el almacenamiento encriptado de las contraseñas de usuarios utilizando el estándar AES256-CBC, de tal forma que son inaccesibles incluso para el administrador de la base de datos.

HTTPS: El protocolo HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible que el protocolo HTTP. Este protocolo además de brindar seguridad, también es un requerimiento que tienen navegadores como Google Chrome para traficar cierta información como la de geolocalización, que utiliza la API de Google Maps.

[4] Fuente: https://www.owasp.org/index.php/Session_fixation
[5] Fuente: https://www.owasp.org/index.php/Clickjacking
[6] Fuente: https://www.owasp.org/index.php/Cross- Site_Request_Forgery_(CSRF)

Métricas:

Creemos importante definir métricas para poder medir que nuestro sistema desarrollado cumple con todos los requerimientos no funcionales previamente definidos. En la siguiente tabla mostramos cuales fueron las que nosotros definimos:

Atributo	Fuente	Estímulo	Artefacto	Entorno	Respuesta	Medida
Modificabilidad	Desarrollador	Agregar nueva validación sobre atributo de una entidad	ABM Usuario	Fase de desarrollo	Validación integrada	Número de clases modificadas: 1
Un desarrollador podrá agregar o cambiar una validación sobre la entidad Usuario. Modificando máximo 1 clase. El cambio impacta en todos los roles.						
Modificabilidad	Desarrollador	Agregar regla de negocio	Realizar pedido	Fase de desarrollo	Regla integrada	Número de clases modificadas: 1
Un desarrollador podrá agregar una nueva regla de negocio al módulo de realizar pedido. Modificando máximo 1 clase. El cambio impacta en todos los roles.						
Seguridad	Usuario	Intentos erróneos consecutivos de acceso	Autenticación de Usuario	Producción	Restablecer contraseña	Número de intentos: 5
El sistema restablecerá la contraseña de una cuenta luego de 5 intentos consecutivos erróneos de ingreso al sistema						
Seguridad	Usuario	Tiempo expiración sesión de usuario.	Autenticación de Usuario	Producción	El sistema redirige a la página de autenticación	Tiempo de sesión: 1800000 milisegundos (30 minutos)
El sistema expirará la sesión de un usuario a los 30 minutos de haber iniciado la misma.						
Escalabilidad	Administrador del sistema	Falta de capacidad de procesamiento de solicitudes HTTP	Servidor Web Amazon	Producción	Nueva instancia corriendo en paralelo	Nueva instancia levantada en 30 minutos
Al dar aviso una alarma, previamente configurada, el administrador del sistema replicará una instancia del servidor web en menos de 2 horas.						

Tabla 4-1 Métricas

Vista lógica:

Esta vista modela elementos lógicos que soportan la funcionalidad que el sistema provee al usuario final desde un punto de vista estático.

El servidor web adopta un modelo basado en MVC (modelo-vista-controlador) en principal medida porque viene heredado por el framework utilizado (AngularJS), las ventajas que posee son que permite estructurar el proyecto de una manera ordenada, contribuyendo a la mantenibilidad del mismo:

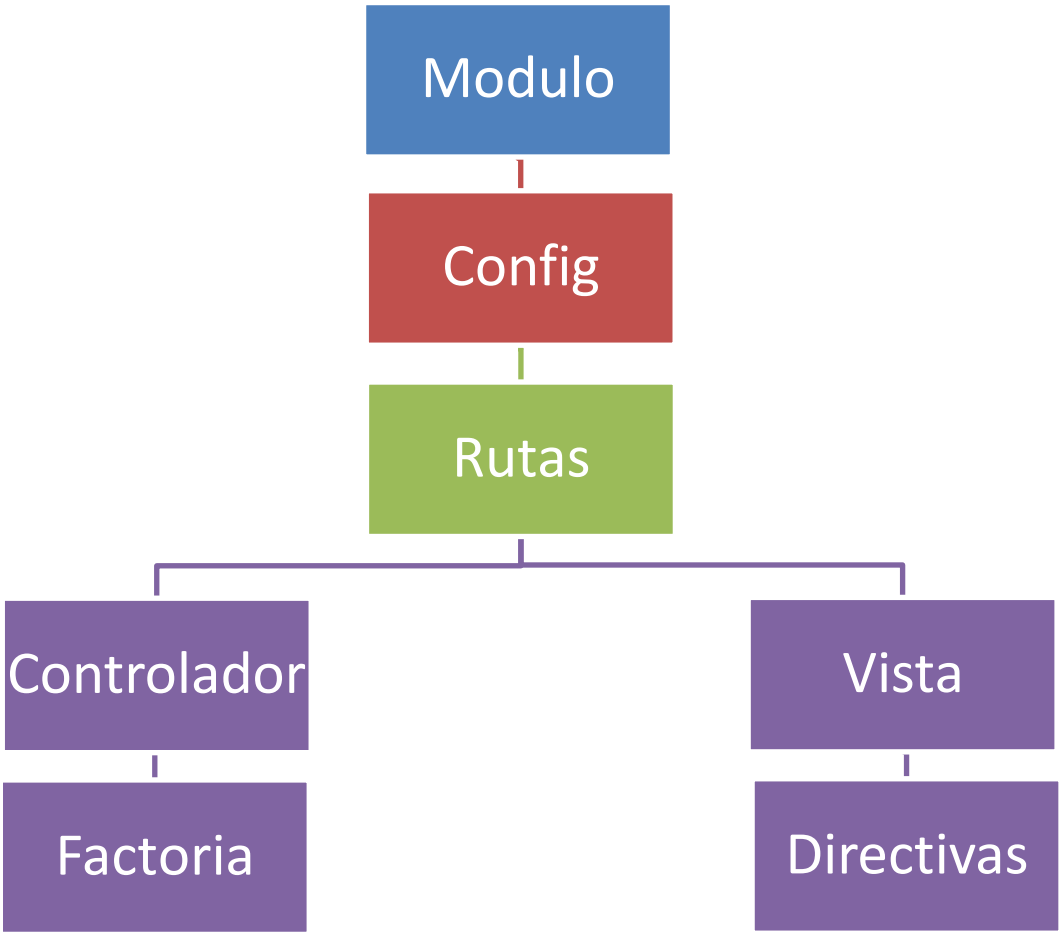

Figura 4-3 Vista lógica (Servidor Web) del sistema

1. La aplicación se puede dividir en distintos módulos, que a su vez pueden estar compuestos de otros módulos. Esta división permite separar distintos comportamientos, así como definir componentes reutilizables, también facilita las pruebas de unidad, que se pueden definir a nivel de módulo.
2. Las factorías son las “recetas” que utiliza AngularJS para instanciar los objetos que realizan todas las operaciones de la aplicación web.
3. Los controladores son los encargados de utilizar factorías para obtener datos, procesarlos y asociarlos a la vista a través del $scope.
4. El $scope (también llamado ViewModel) contiene toda la información que se va a mostrar en las vistas.
5. Las vistas se enlazan al $scope que viene desde su controlador. Esta asociación se hace a través de componentes llamadas directivas.
6. Mas explícitamente, las directivas son una expansión del lenguaje html, que le permite realizar bloques de control, manejar eventos y asociar información con el $scope.

Este servidor web se va a conectar al servidor de aplicación mediante Web API REST, las cuales serán definidas en la capa superior del servidor de aplicación. Los servicios se definen de tal manera que sean independientes del cliente que los consume, esto permite que más adelante se pueda agregar una plataforma móvil, sin modificar en absoluto nuestro servidor de aplicación.

El servidor de aplicación está estructurado en mediante un modelo de capas que representan el tratamiento que tendrá la información que llega al servidor:

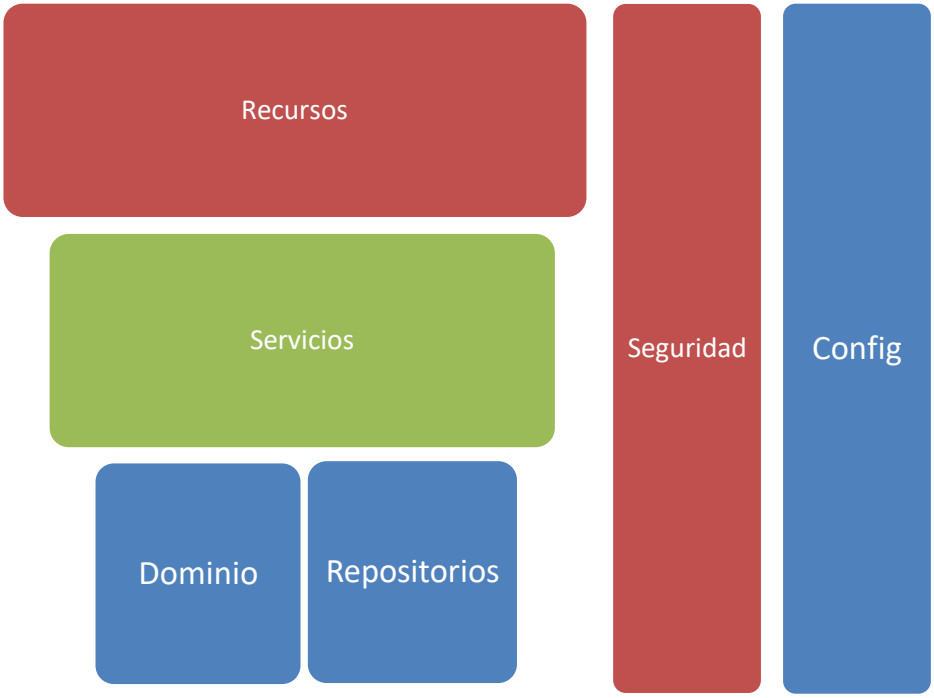

Figura 4-4 Vista lógica (Servidor de aplicación) del sistema

.

1. Recursos: Expone las Web API que toman las llamadas del servidor web, realizan las acciones necesarias y devuelven la respuesta.
2. Servicios: Definen las funcionalidades principales del sistema, pueden ser llamados desde los recursos o desde otros servicios.
3. Dominio: Define la lógica de negocio, es decir el modelo de clases y sus correspondientes atributos.
4. Repositorios: Implementan las operaciones para recuperar y guardar información sobre la instancia de MongoDB.

5. Seguridad: Implementa todas las características de seguridad del servidor de aplicación.
6. Config: contiene todas las configuraciones que necesitan las distintas funcionalidades de la aplicación para funcionar, ya sea registro de operaciones, conexión a la base de datos, paquetes utilizados, opciones de mapeo, configuración de ejecución, etc.

Este modelo permite mantener separados estos tres aspectos del tratamiento de la información, y permitir que puedan ser modificados sin afectar al resto en tanto se respeten sus interfaces de comunicación. De esta forma se ayuda a disminuir el acoplamiento entre componentes.

5. TECNOLOGÍAS Y ENTORNO DE DESARROLLO

A continuación se describen las principales tecnologías, plataformas y herramientas que se utilizaron durante el desarrollo del proyecto.

Como base central utilizamos JHipster, que es una herramienta novedosa y muy innovadora para crear aplicaciones web. Esta herramienta se basa en un back-end Java (Spring Boot) y un front-end Angular. Y un conjunto de módulos adaptables a nuestra aplicación en lo que refiere a bases de datos, seguridad, pruebas, servidores, etc.

Figura 5-1 Logo framework JHipster

Para la generación de código, JHipster, se basa en Yeoman, que es otra importante herramienta, la cual permite crear cualquier tipo de aplicación en cualquiera de los lenguajes de programación más utilizados del momento (Java, Python, C #, etc.)

Figura 5-2 Logo Yeoman

En base a la estructura que propone JHipster, realizamos una descripción de los principales frameworks y tecnologías que utilizamos por cada área:

Front-end:

- **HTML5:** es un lenguaje markup usado para estructurar y presentar el contenido para la web.
- **JavaScript:** es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- **CSS:** es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas.

Figura 5-3 Logo lenguajes HTML5, JavaScript y CSS3

- **Frameworks:**
    - **AngularJS:** es un framework Javascript basado en MVC (Modelo-Vista-Controlador) y que se centra en intentar dinamizar documentos HTML, DHTML (Dynamic HTML) y ejecutarse como aplicación de una sola página.

Figura 5-4 Logo framework AngularJS

Back-end:

- **Java:** es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más.

Figura 5-5 Logo Java

- IDE:
  - **Eclipse:** es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plug-ins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo.

Figura 5-6 Logo entorno de desarrollo Eclipse

- Frameworks:
    - **Spring:** es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

Figura 5-7 Logo framework Spring

    - **Hibernate:** es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Figura 5-8 Logo framework Hibernate

Base de Datos:

MongoDB: es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Figura 5-9 Logo base de datos MongoDB

Pruebas:

Decidimos utilizar herramientas que abarquen distintos tipos de pruebas. Y en conjunto con los test unitarios que provee Spring y Maven, describimos a continuación las tecnologías aplicadas:

KarmaJS: herramienta para realizar pruebas de interfaz de usuario. Desarrollado y mantenido por la comunidad de código abierto en GitHub.

Figura 5-10 Logo herramienta de pruebas Karma

Gatling: herramienta para realizar pruebas de performance/rendimiento. Aplicable a cualquier servidor HTTP.

Figura 5-11 Logo herramienta de pruebas Gatling

Protractor: herramienta para realizar pruebas de integración para aplicaciones Angular.

Figura 5-12 Logo herramienta de Protractor

Elegimos estas herramientas por su facilidad de uso, facilidad de mantenimiento y alto rendimiento.

Servidor:

Amazon AWS EC2: es un servicio web que proporciona capacidad de cómputo con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores el cloud computing escalable basado en web. Reduce el tiempo necesario para obtener y arrancar nuevas instancias en cuestión de minutos, lo que permite escalar rápidamente la capacidad, ya sea aumentando o reduciéndose, según cambien las necesidades.

Figura 5-13 Logo amazon EC2

Apache2: El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.) El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la

supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Para el proyecto configuramos un servidor Apache como front-end para ocultar nuestro servidor en el que realmente está corriendo la aplicación. Esta configuración es conocida con el nombre de “proxy reverso”. Este proxy acepta las solicitudes de los usuarios de internet, y redirige el tráfico a nuestro servidor web.

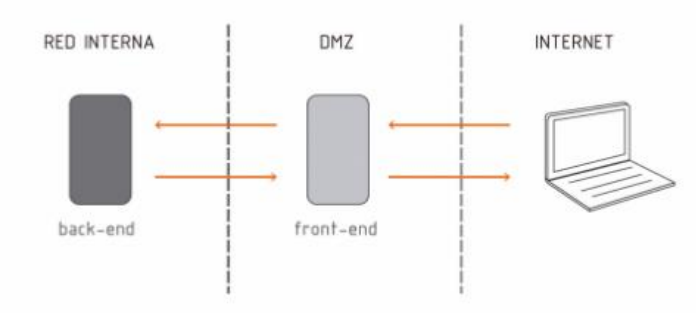

Figura 5-14 Funcionamiento proxy reverso Apache2

Nombramos algunos de los beneficios que obtenemos:

- **Seguridad:** podemos decir que al configurar un proxy reverso, agregamos una capa más de seguridad a nuestros servidores web, primero porque ocultamos la topología y las características de los servidores detrás del proxy eliminando el acceso directo desde internet hacia ellos.
- **Balanceo de Carga y Failover:** un proxy reverso nos puede servir además como balanceador de carga, recibe todas las solicitudes de los usuarios y los distribuye en varios servidores para que toda la carga no vaya a uno solo.
- **Simplifica el Control de Acceso:** al crear un único punto de acceso, podemos concentrar todos nuestros esfuerzos en ese punto indicando un rango de ip que no queremos que haga solicitudes a nuestros servidores web.
- **Varios sitios web en una misma url:** podemos tener varias aplicaciones distribuidas en diferentes servidores. Un proxy reverso puede

enrutar varias ramas de la URL a distintos servidores web internos (configurando distintos Virtual Host). Para el usuario está navegando en el mismo servidor pero realmente está viendo contenido de distintos servidores.

Google APIs: es un conjunto de interfaces de programación de aplicaciones (APIs) desarrollado por Google que permiten la comunicación con los servicios de Google y su integración con otros servicios.

Figura 5-15 Logo Google APIs

Utilizamos estos servicios para georreferenciar, geo localizar y ubicar en Maps a los negocios y usuarios de la aplicación. Nos son de gran utilidad y dependemos de los mismos porque realizamos las búsquedas en base a las coordenadas del usuario.

Para su utilización creamos una clave api para el proyecto y con ella accedemos a una cuota gratuita de uso diariamente. A continuación mostramos un gráfico del tráfico de solicitudes realizadas:

Figura 5-16 16 Gráfica de las peticiones hechas por PediRapido hacia Google APIs

Seguido de una tabla que indica que librerías fueron utilizadas, con la cantidad de solicitudes correctas, con error y el porcentaje de los mismos:

API	Solicitudes	Errores	Proporción de errores
Google Maps JavaScript API	9	0	0 %
Google Maps Geocoding API	7	0	0 %
Google Maps Directions API	–	–	–
Google Maps Distance Matrix API	–	–	–
Google Maps Elevation API	–	–	–
Google Maps Geolocation API	–	–	–

Figura 5-17 Tabla de las distintas librerías solicitadas por PediRapido hacia Google APIs

Herramientas que darán soporte al desarrollo:

- **Google Drive:** herramienta de edición colaborativa de documentos en tiempo real. Utilizamos esta herramienta como repositorio de archivos debido a la facilidad de acceso y edición por parte de los integrantes del proyecto.

Figura 5-18 Logo Google Drive

Aquí tenemos almacenados la descripción de las funcionalidades, las historias de usuario, y demás documentos. El plan de las iteraciones se implementó en una hoja de cálculo para ir asignando historias de usuario a cada desarrollador, describiendo su trabajo, fechas de comienzo y de fin,

archivos afectados del proyecto y, en el caso de que exista, una descripción. A continuación mostramos la tabla diseñada:

4	Cod. Hist.	Nombre Historia	Estado	Desarrollador	Fecha Inicio	Fecha Fin
	C.3.2	Ingreso automático	terminada	G.O	9/02/2016	11/02/2016
	C.5.1	Filtrar búsqueda por orden alfabético	terminada	G.O	4/03/2016	4/03/2016
	C.5.2	Filtrar búsqueda por cercanía	terminada	G.O	5/03/2016	7/03/2016
	C.7.1	Reintentar pedido	terminada	S.G	20/03/2016	2/04/2016

Figura 5-19 Tabla utilizada para el registro de trabajo en las historias de usuario

En la etapa de prueba, al finalizar cada iteración, surgieron distintas mejoras y aparecieron bugs, por lo que decidimos crear otra pestaña y la documentamos también con el desarrollador asignado, estado, fechas de inicio y fin. A continuación mostramos un fragmento de esta tabla:

MEJORAS Y BUGS					
Descripcion	ugar (Clase-Proyecto- el	Estado	Desarrollador	Fecha Inicio	Fecha Fin
Cambiar el modal de seleccionar cantidad de un producto a elegir por uno	Comidas.html	terminada	G.O	13/02/2016	15/02/2016
Incorporar a Negocio las entidades Comentario	Backend	terminada	G.O		
Modal para seleccionar cantidad, cambiar el seleccionador	Comidas.html	terminada	G.O	28/03/2016	28/03/2016
Pantalla principal User	main.html	terminada		18/05/2016	13/06/2016
pantalla principal Proveedor	mainProveedor.html	terminada	S.G		
cargar todas las imagenes desde mongo	-	terminada	G.O		15/05/2016
no aparece la direccion al buscar automaticamente y limpiar el formato	main.html	terminada	G.O	14/05/2016	1/06/2016
agregar tabs para distintos estados de pedido	mainProveedor.html	terminada	S.G	3/04/2016	3/04/2016
agregar modif de precio en disp de producto	mainProveedor.html	terminada	S.G	2/04/2016	11/04/2016

Figura 5-20 Tabla utilizada para el registro de trabajo en las mejoras y bugs

Para las validaciones correspondientes a cada historia, documentamos en una nueva tabla donde agrupamos las mismas en base al rol al que pertenecía, junto con una breve descripción y la sección para una mejor identificación. A continuación mostramos un fragmento de esta tabla:

Rol	Seccion	Validacion	Desarrollador	Estado
USER	carga de direccion	ciudad: (obligatorio), direccion (obligatorio tam max:50)	SG	terminada
USER	seleccion producto	cantidad: numeros 0<x<500	SG	terminada
USER	lineas de pedido	catidad max: 20	SG	terminada
USER	btn confirmar pedido	cantidad lineas de pedido >0	SG	terminada
USER	confirmacion pedido	cuanto paga: (numeros 0<x<100000), comentario: (tam	G.O	terminada
USER	resumen/carga de direccion	ciudad: (obligatorio), direccion (obligatorio tam max:50)	SG	terminada

Figura 5-21 Tabla utilizada para el registro de las validaciones realizadas

- **SVN:** herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja al de un sistema de ficheros. El repositorio donde alojamos la aplicación es RiouxSVN, que es totalmente gratuito y privado (https://riouxsvn.com/)

6.EXTENSIBILIDAD

Al finalizar el desarrollo de este informe, el proyecto se encuentra con las 8 iteraciones con sus etapas de diseño, desarrollo, prueba y despliegue completamente finalizadas. La aplicación se encuentra publicada en un servidor.

Se exponen a continuación algunos puntos relacionados a la extensibilidad del proyecto:

- Es tendencia mundial el crecimiento del uso y de las compras desde dispositivos móviles, por eso consideramos que uno de los caminos posibles en este contexto es migrar el actual sistema a una aplicación móvil para dispositivos con android y iOS, utilizando en Apache Cordova para realizar esta conversión. Todo esto con el objetivo de que los usuarios y los proveedores tengan más facilidades para acceder a nuestros servicios.
- Ofrecer la posibilidad de pagar el pedido mediante las plataformas actuales de pago online. Esto brindará mayor seguridad de ambos lados e implicaría una gran innovación en el tema, ya que estas plataformas de pago se pueden integrar completamente a la aplicación manejada por los negocios.
- Personalizar la búsqueda de negocios y de ofertas para los usuarios, basándonos en sus pedidos realizados, los días de la semana y en la hora que se realizaron.
- Ofrecer un seguimiento del pedido más detallado, utilizando GPS para saber exactamente la ubicación del mismo una vez que fue despachado por el negocio.

7. CONCLUSIONES

Se presentan a continuación aspectos considerados más relevantes que obtuvimos al finalizar el proyecto final de carrera.

a. Acerca de la planificación

Creemos importante destacar que el alcance definido en la planificación del proyecto fue realista y alcanzable. Cumplimos con la totalidad de las historias definidas en cada una de las 8 iteraciones, también con mejoras que agregamos, las mismas fueron surgiendo en cada una de las etapas.

A la hora de la definición de las historias de usuarios consideramos que, por la falta de experiencia con estas tecnologías y por ser el primer proyecto de este tamaño que afrontamos, hubo historias que tendríamos que haber sido más precisos en su definición. De estos casos derivamos a nuevamente documentar mejoras porque quedan temas sin definir. Definimos casos felices y no tuvimos en cuenta flujos auxiliares. Más allá de esto, siendo nosotros los que definimos las funcionalidades del sistema, creemos que el número de historias y a lo que apuntaba cada una de ellas estuvo muy bien.

En lo que respecta a la estimación del proyecto, creemos que definimos adecuadamente la velocidad del equipo, ya que consideramos muchas falencias que teníamos en cuanto a conocimientos técnicos y de gestión. Además de ser realistas en cuanto a nuestra disponibilidad de tiempo con la que íbamos a contar.

Hubo historias donde no tuvimos en cuenta la reutilización de componentes y de servicios, lo que nos ahorró tiempo que habíamos destinado al desarrollo. Pero otros aspectos, como cuestiones de diseño gráfico y web nos costaron más de lo pensado.

b. Acerca del desarrollo

De todas las etapas realizadas en el proyecto nos llevamos cuestiones muy positivas. Fundamentalmente la experiencia adquirida, ya que realizamos muchas de estas actividades por primera vez. Por supuesto que serán tenidas en cuenta para nuestros siguientes proyectos y es esto lo que realmente vale.

Para este proyecto utilizamos tecnologías de última generación, tuvimos que formarnos en las mismas y, lo más importante de todo, nos deja puertas abiertas para seguir investigando sobre las nuevas que van surgiendo ya que están en continuo avance.

Por lo anteriormente propuesto, y por haber alcanzado los objetivos planteados tanto en funcionalidades, como en tiempo concluimos que la elección de la metodología fue acertada y su implementación exitosa.

A modo de conclusión final, para este proyecto aplicamos distintos conceptos, metodologías y conocimientos, los cuales nos fueron enseñados durante nuestro paso por la facultad, que si bien no siempre son aplicables directamente, sirven como base para aprender sobre cualquier aspecto de la ingeniería. Sumado esto a todo lo aprendido en el transcurso del proyecto y lo que nos queda por aprender consideramos positiva esta nueva experiencia con la cual concluye una hermosa etapa en nuestras vidas.

8.GLOSARIO

Se enuncian a continuación los principales conceptos manejados en este proyecto y la definición correspondiente:

- **Cliente Final:** es el usuario que representa aquellas personas que realizan los pedidos.

- **Cliente Proveedor:** es el usuario que representa algún negocio y ofrece pedidos para los clientes finales.

- **Administrador:** es el usuario que se encarga de gestionar clientes proveedores (con sus negocios asociados) y clientes finales.

- **XP (extreme programing):** es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck. Es el más destacado de los procesos ágiles de desarrollo de software. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

- **Story (Historia):** una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Son utilizadas en las metodologías ágiles para la especificación de requisitos. Cada historia de usuario debe ser limitada, ésta debería poderse escribir sobre una nota pequeña.

- **Story Point:** punto de la historia es una medida arbitraria utilizado por equipos de Scrum. Esto se utiliza para medir el esfuerzo requerido para implementar una historia de usuario.

● **Task (Tarea):** representa una actividad atómica que realiza una persona o grupo de trabajo en un tiempo delimitado, la suma de tareas contribuye a la realización de una historia de usuario.

● **Task Point:** es una unidad de medida del esfuerzo de realización de una tarea. Se utiliza para medir el avance sobre la realización de tareas.

● **Front-end:** es la parte del software que interactúa con el o los usuarios. Se encarga de maquetar la estructura semántica del contenido (HTML), codificar el diseño en hojas de estilo (CSS) y agregar la interacción con el usuario (Javascript).

● **Back-end:** es la parte que procesa la entrada desde el front-end. Se encarga de implementar bases de datos como MySQL, Postgres, SQL Server o MongoDB. Luego, un lenguaje como PHP o Java, o frameworks como Django, Node.JS o .NET se conectan a la base de datos.

● **Servidor Web:** Un servidor web o servidor HTTP es software que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente (HTTP en nuestro caso). El código recibido por el cliente suele ser compilado y ejecutado por un navegador web.

● **Servidor de aplicación:** usualmente se trata de un software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación.

● **MVC:** el modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

● **Framework:** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

● **IDE:** es una aplicación informática que proporciona servicios integrales para facilitar al programador el desarrollo de software, las herramientas que normalmente posee son un editor de código, depurador, compilador o intérprete.

● **Documento de requerimientos:** es un documento que permite comunicar de forma precisa los requerimientos, objetivos y presunciones del proyecto. Sirve como un contrato con partes externas y como herramienta de estimación. También se utiliza como referencia a la hora de evaluar el resultado el proyecto, y como último se utiliza como base a la hora de realizar mantenimiento al sistema terminado.

● **Requerimientos funcionales:** un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas.

● **Requerimientos no funcionales:** un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar.

● **Despliegue:** son todas las actividades que hacen que un sistema o aplicación de software esté disponible para su uso.

9. REFERENCIAS BIBLIOGRÁFICAS

Yampier Medina Tarancón. (2012).Una taxonomía para la identificación de riesgos en los proyectos de desarrollo de software de la Universidad de las Ciencias Informáticas. Universidad de las Ciencias Informáticas. La Habana, Cuba.

http://publicaciones.uci.cu/index.php/SC/article/view/785/511

Wikipedia, Desarrollo ágil de software

https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software

Kent Beck. (2002). Una explicación de la programación extrema: aceptar el cambio. Addison-Wesley.

Azaustre, Carlos. ¿Qué es AngularJs?

https://carlosazaustre.es/blog/empezando-con-angular-js/

Página oficial de MongoDB

https://www.mongodb.com/es

Página oficial de Jhipster

https://jhipster.github.io/

Página oficial de yeoman

http://yeoman.io/

Página oficial de AWS

https://aws.amazon.com/es/ec2/

Página oficial de Spring

https://spring.io/

Página oficial de Bootstrap

http://getbootstrap.com/

Documentación Google APIs Console

https://console.developers.google.com/apis/

10. ANEXO

a. Manual de instalación

Descarga e instalación Java 8

- Descargar el .exe para 32 o 64 bits (dependiendo el sistema operativo) desde este enlace: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
- Ejecutar el archivo descargado y seguir las instrucciones. Indicar la ruta destino, recomendamos la ruta por defecto seleccionada (C:\Program Files (x86)\Java).
- Si no se establece, setear variable del sistema JAVA_HOME.
    - En Windows:

- Panel de Control -> Sistema y seguridad -> Sistema -> Configuración avanzada del sistema.

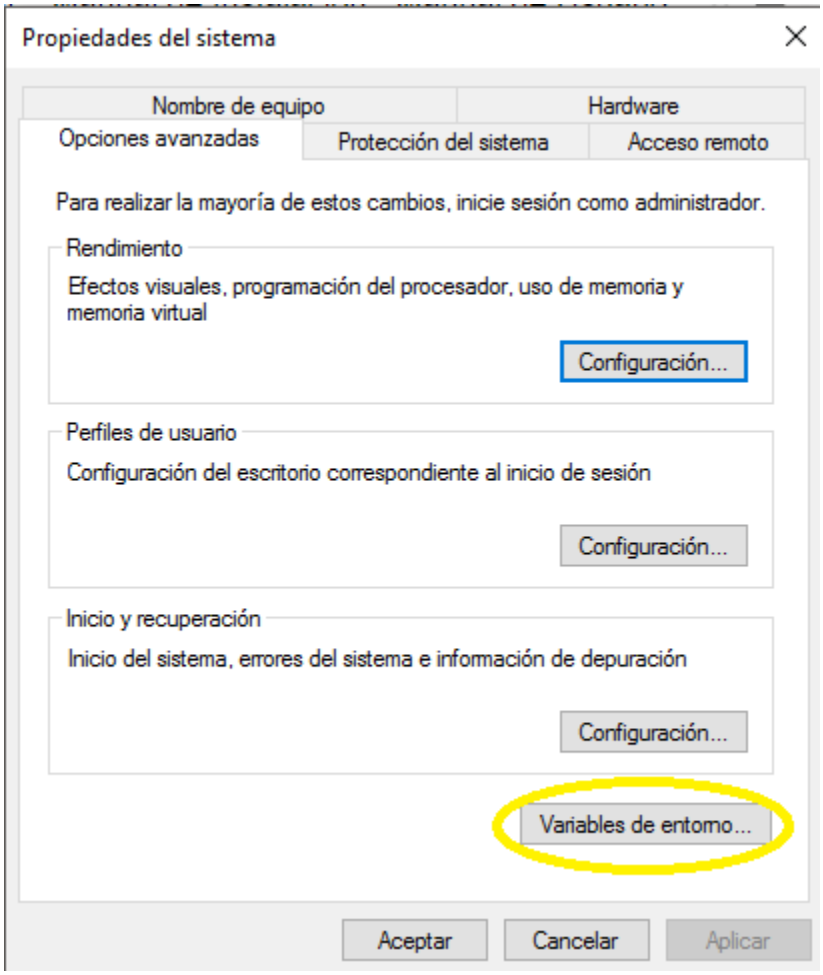

Figura 10-1 Variables de entorno del sistema en Windows

- Ingresar a Variables de entorno y luego setear la ruta en la que se instaló Java en la variable de sistema JAVA_HOME (crearla en caso de que no exista)

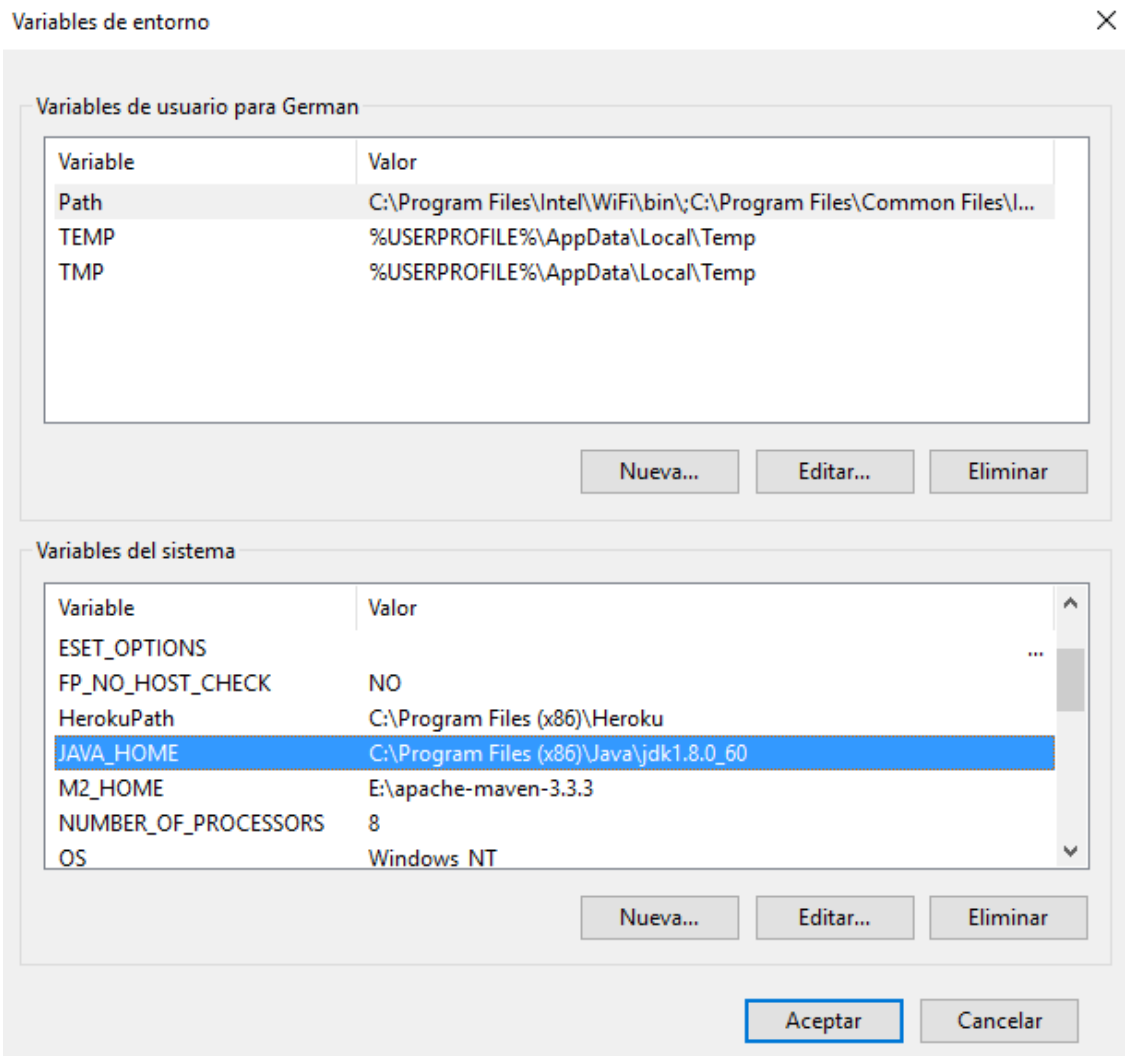

Figura 10-2 Edición de variable de entorno JAVA_HOME

- Por último verificar la versión de Java desde la consola de comandos.

Ejecutar el siguiente comando: *java -version*

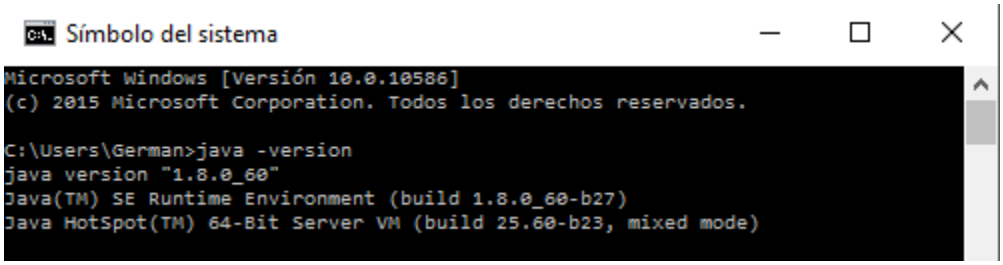

Figura 10-3 Versión instalada de Java desde la consola de comandos

Descarga e instalación de MongoDB

- Descarga desde la página oficial de Mongo (32 o 64 bits) desde https://www.mongodb.com/download-center#community
- Lo descomprimimos y seguimos las instrucciones.
- Crear un directorio llamado “mongodb” en el directorio raíz “C:”. Aquí irá todo el contenido descomprimido.
- Luego, en el disco C: creamos el directorio “data” y dentro de este otro directorio llamado “db”, aquí Mongo almacenará la información de las bases de datos.
- Para que funcione como un servicio crear el directorio “log” dentro de “C:/mongodb/”.
- Ejecutar el siguiente comando:
  - $ echo logpath=C:\mongodb\log\mongo.log > C:\mongodb\mongod.cfg
- Luego instalamos el servicio:
  - $ C:\mongodb\bin\mongod.exe --config C:\mongodb\mongod.cfg --install

- Por último, para dejar el servicio corriendo ejecutar este archivo:

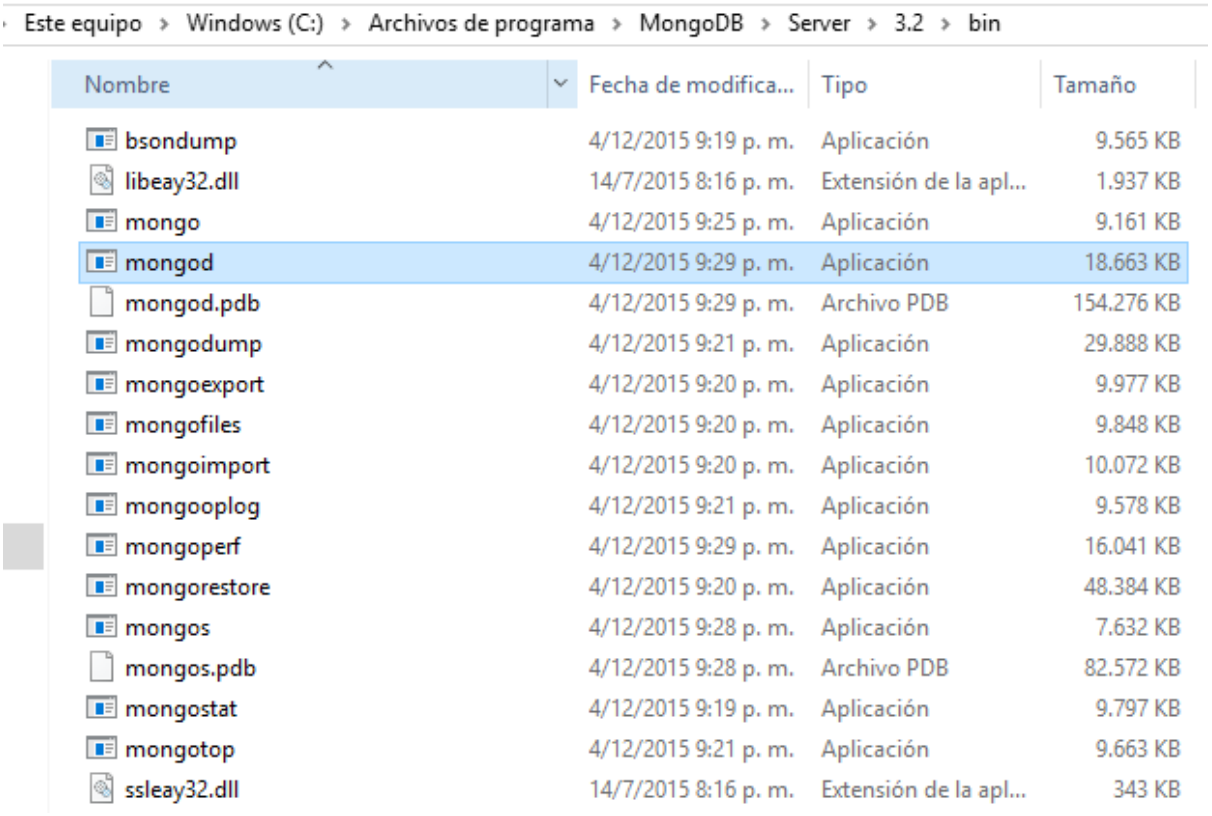

Figura 10-4 Directorio de instalación de MongoDB

- Y veremos en la consola:

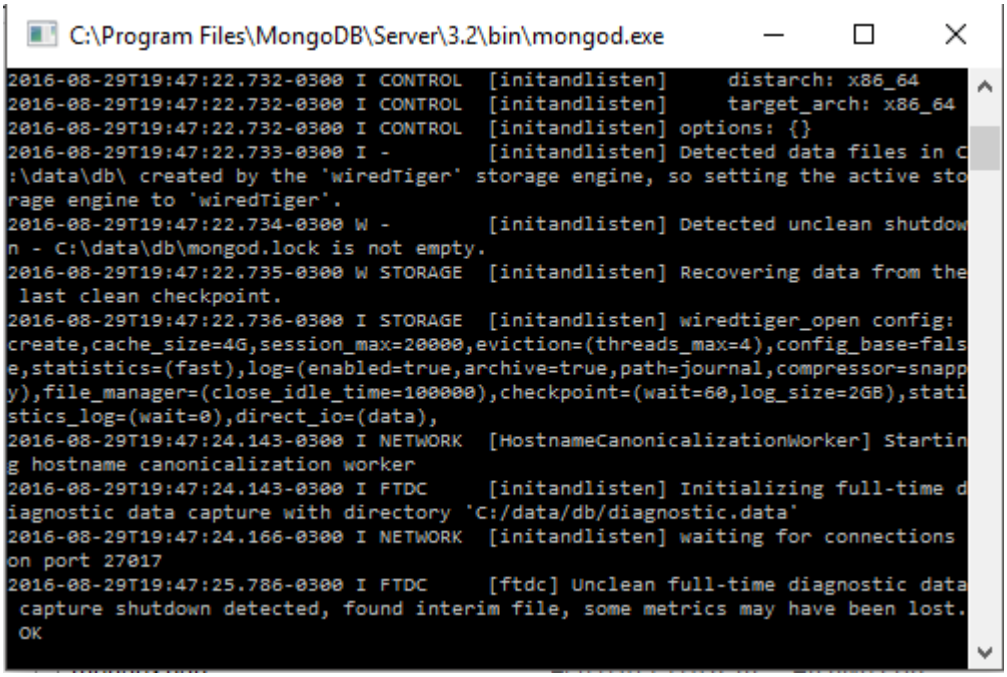

Figura 10-5 Iniciando MongoDB

Más información: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

Por último, una vez que tenemos instalado Java JDK 8 y tenemos corriendo MongoDB en el puerto 27017 solicitar el archivo pedi-rapido-0.0.1-SNAPSHOT.war (ejecutable) del proyecto a alguno de los integrantes.

Posicionarse en la consola de comandos en el directorio donde se tiene el .war y ejecutar el siguiente comando:

java -jar pedi-rapido-0.0.1-SNAPSHOT.war --spring.profiles.active=dev

Y la aplicación comenzará a cargar:

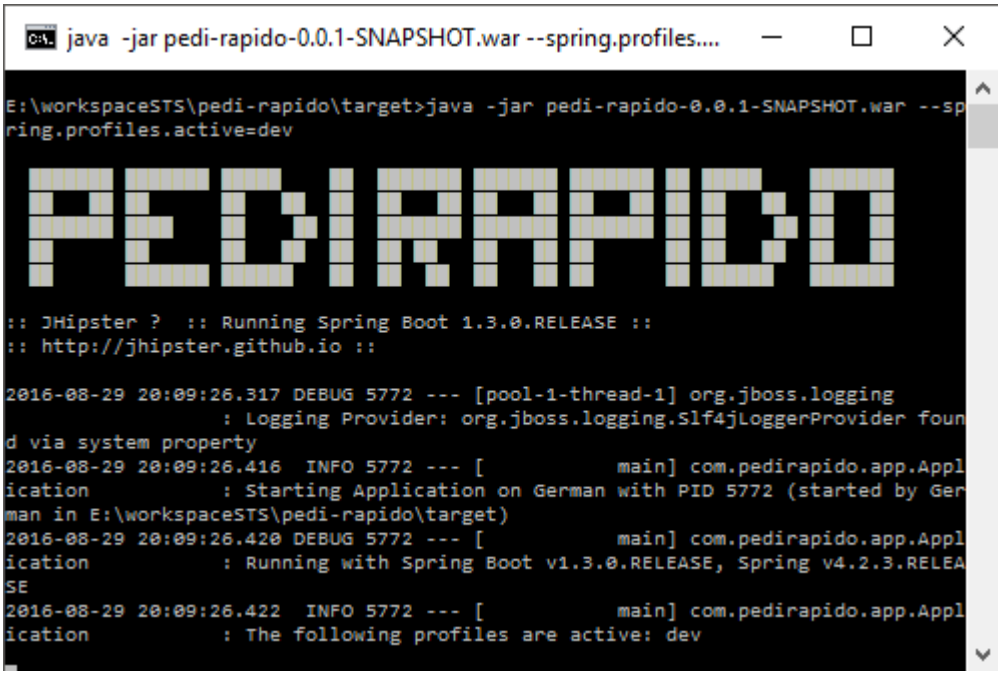

Figura 10-6 Despliegue aplicación PediRapido

Por último, veremos que esta online en el puerto 8210 (puerto configurado para la instancia de desarrollo):

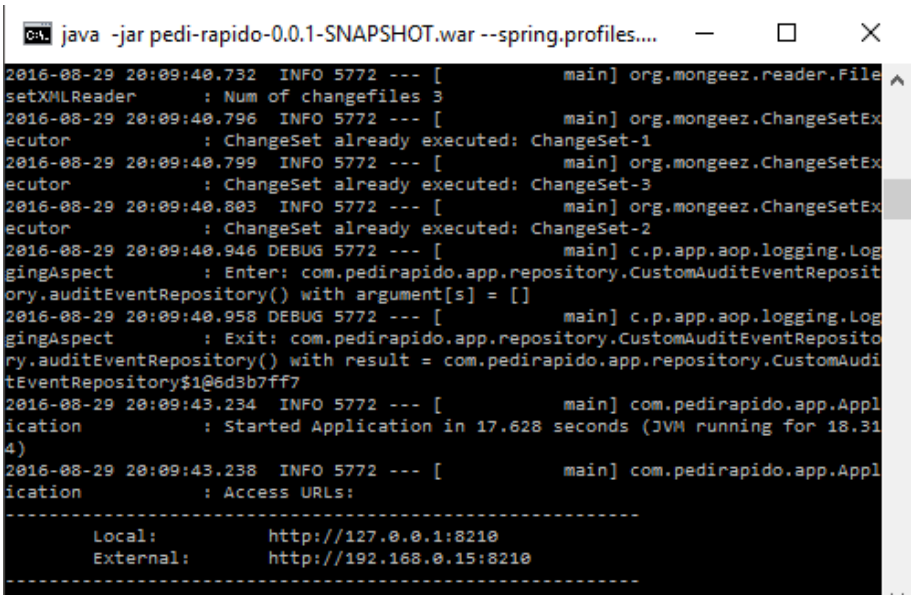

Figura 10-7 Despliegue exitoso

b. Pruebas funcionales

A continuación se muestra la planilla con la resolución de todos los casos de prueba planteados:

Nro.	Historia	Usuario	Caso de prueba	Resultado esperado	Resultado obtenido
1	C.1 Inicio de sesión de usuario	Cliente final	el usuario intenta iniciar sesión con un usuario y contraseña válidos	se inicia la sesión correctamente y se redirige al usuario a la pantalla principal	**aprobado**
2	C.1 Inicio de sesión de usuario	Cliente final	el usuario intenta iniciar sesión con un usuario válido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
3	C.1 Inicio de sesión de usuario	Cliente final	el usuario intenta iniciar sesión con un usuario inválido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
4	C.1 Inicio de sesión de usuario	Cliente final	el usuario intenta iniciar sesión con un usuario válido y sin contraseña	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
5	C.2 Alta de usuario	Cliente final	el usuario intenta registrarse con nombre, email, contraseña y confirmación válidos	el sistema indica que se ha registrado la cuenta y envía el mail de confirmación	**aprobado**
6	C.2 Alta de usuario	Cliente final	el usuario intenta registrarse con un nombre ya ocupado	el sistema indica que el usuario indicado ya se encuentra en uso	**aprobado**
7	C.2 Alta de usuario	Cliente final	el usuario intenta registrarse con un mail ya ocupado	el sistema indica que el mail indicado ya se encuentra en uso	**aprobado**
8	C.2 Alta de usuario	Cliente final	el usuario intenta registrarse con una	el sistema indica que debe ingresar una	**aprobado**

			contraseña inválida	contraseña valida	
9	C.2 Alta de usuario	Cliente final	el usuario intenta registrarse con una confirmación de contraseña inválida	el sistema indica que debe ingresar una confirmación de contraseña valida	**aprobado**
10	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección válida ingresada manualmente	el sistema guarda la dirección y pasa a la sección de negocios	**aprobado**
11	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección inválida ingresada manualmente	el sistema indica que la dirección ingresada no es valida	**aprobado**
12	C.3 Indicar ubicación	Cliente final	El usuario intenta ingresar con una dirección válida ingresada manualmente y piso y depto. válidos	el sistema guarda la dirección y pasa a la sección de negocios	**aprobado**
13	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección válida ingresada automáticamente	el sistema guarda la dirección y pasa a la sección de negocios	**aprobado**
14	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección válida predeterminada	el sistema guarda la dirección y pasa a la sección de negocios	**aprobado**
15	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección válida de formato pero que no existe	el sistema indica que no puede encontrar dicha dirección y no permite continuar	**no aprobado**
16	C.3 Indicar ubicación	Cliente final	el usuario intenta ingresar con una dirección vacía	el sistema indica que la dirección es requerida	**aprobado**
17	C.4 Elegir rubro	Cliente final	el usuario ingresa al rubro "Comidas"	se muestra la pantalla para cargar la dirección, y luego de ingresarla se muestran los negocios del rubro	**aprobado**

				"comidas"	
18	C.4 Elegir rubro	Cliente final	el usuario ingresa al rubro "Heladerías"	se muestra la pantalla para cargar la dirección, y luego de ingresarla se muestran los negocios del rubro "heladerías"	**aprobado**
19	C.4 Elegir rubro	Cliente final	el usuario ingresa al rubro "Bebidas"	se muestra la pantalla para cargar la dirección, y luego de ingresarla se muestran los negocios del rubro "bebidas"	**aprobado**
20	C.4 Elegir rubro	Cliente final	el usuario ingresa a un rubro luego vuelve al inicio y elige otro rubro	el sistema muestra correctamente los negocios del último rubro elegido	**aprobado**
21	C.5 Elegir comercio	Cliente final	el usuario oprime el botón para ordenar alfabéticamente	los negocios se muestran ordenados alfabéticamente	**aprobado**
22	C.5 Elegir comercio	Cliente final	el usuario oprime el botón para ordenar por cercanía	los negocios se muestran ordenados por cercanía	**aprobado**

23	C.5 Elegir comercio	Cliente final	el usuario selecciona un elemento de la lista de tipos de comidas	solo se muestran negocios que contienen el tipo de comida seleccionado	**aprobado**
24	C.5 Elegir comercio	Cliente final	el usuario oprime el botón para ordenar por popularidad	los negocios se muestran ordenados por popularidad	**aprobado**
25	C.6 Armar el pedido	Cliente final	el usuario clickea para desplegar y plegar todas las categorías del negocio	para todas las categorías los productos se muestran correctamente	**aprobado**
26	C.6 Armar el pedido	Cliente final	el usuario selecciona un producto, selecciona una cantidad valida y acepta	el producto se agrega al pedido con la cantidad ingresada y el precio correspondiente	**aprobado**
27	C.6 Armar el pedido	Cliente final	el usuario selecciona un producto, selecciona una cantidad valida y cancela	el producto no se agrega al pedido y al elegir otro se restablece el valor predeterminado	**aprobado**
28	C.6 Armar el pedido	Cliente final	el usuario selecciona un producto, selecciona una cantidad mayor a la permitida y acepta	el sistema indica que se superó la cantidad máxima y no permite aceptar	**no aprobado**
29	C.6 Armar el pedido	Cliente final	el usuario selecciona un producto, selecciona una cantidad menor a la permitida y acepta	el sistema no permite elegir una cantidad menor al mínimo	**aprobado**

30	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar con un pedido vacío	el sistema indica que no se puede confirmar un pedido vacío	**aprobado**
31	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar con un pedido con más ítems de los permitidos	el sistema indica que no se puede confirmar un pedido con más ítems de los permitidos	**aprobado**
32	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar con un pedido con ítems válidos	el sistema muestra la pantalla de confirmación de pedido	**aprobado**
33	C.7 Confirmar el pedido	Cliente final	el usuario agrega varios ítem, los limpia y hace click en confirmar el pedido	el sistema indica que no se puede confirmar un pedido vacío	**aprobado**
34	C.7 Confirmar el pedido	Cliente final	el usuario agrega in ítem, lo elimina y hace click en confirmar el pedido	el sistema indica que no se puede confirmar un pedido vacío	**aprobado**
35	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar el pedido, ingresa una cantidad invalida en cuanto paga y confirma	el sistema muestra un mensaje de error y no permite realizar el pedido	**aprobado**
36	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar el pedido, ingresa una cantidad valida en cuanto paga y confirma	el sistema realiza el pedido correspondiente	**aprobado**

37	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar el pedido, ingresa un comentario invalido y confirma	el sistema muestra un mensaje de error y no permite realizar el pedido	**aprobado**
38	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar el pedido, ingresa un comentario valido y confirma	el sistema realiza el pedido correspondiente	**aprobado**
39	C.7 Confirmar el pedido	Cliente final	el usuario hace click en confirmar el pedido y confirma sin escribir comentario ni con cuanto paga	el sistema realiza el pedido correspondiente	**aprobado**
40	C.7.1 Reintentar pedido	Cliente final	el usuario hace un pedido y el proveedor no responde luego de un tiempo determinado	el sistema muestra una ventana preguntando si quiere reintentar el pedido	**aprobado**
41	C.7.1 Reintentar pedido	Cliente final	el usuario indica que quiere reintentar el pedido	el sistema sigue comprobando el estado del pedido	**aprobado**
42	C.7.1 Reintentar pedido	Cliente final	el usuario indica que no quiere reintentar el pedido	el sistema abandona el pedido y muestra la pantalla inicial normalmente	**aprobado**
43	C.7.1 Reintentar pedido	Cliente final	el usuario indica que quiere reintentar el pedido 3 veces	el sistema sigue comprobando el estado del pedido	**aprobado**
44	C.8 Resultado del pedido	Cliente final	el proveedor acepta un pedido vigente	el sistema indica que el pedido fue aceptado	**aprobado**
45	C.8 Resultado del pedido	Cliente final	el proveedor acepta un pedido vigente luego de que el usuario haya reintentado	el sistema indica que el pedido fue aceptado	**aprobado**
46	C.8 Resultado del pedido	Cliente final	el proveedor marca el pedido como en camino	el sistema indica que el pedido está en camino	**aprobado**

47	C.8 Resultado del pedido	Cliente final	el proveedor rechaza el pedido	el sistema indica que el pedido fue rechazado	**aprobado**
48	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace click en el botón detalle de un pedido en curso	el sistema muestra el detalle del pedido	**aprobado**
49	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace click en el botón seguimiento de un pedido en curso	el sistema el estado del pedido	**aprobado**
50	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace click en el botón detalle de un pedido en el historial	el sistema muestra el detalle del pedido	**aprobado**
51	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace click en el botón repetir pedido de un pedido en el historial	el sistema muestra el modal para confirmar el pedido con todos los datos correspondientes al pedido elegido	**aprobado**
52	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace click en el botón comentar de un pedido en el historial	el sistema muestra el modal para realizar un comentario sobre el pedido seleccionado	**aprobado**
53	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario confirma un pedido válido a partir de la opción repetir pedido	el sistema realiza el pedido correspondiente	**aprobado**

54	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace un comentario y lo guarda como positivo	el sistema guarda el comentario para ese negocio y no permite comentar ese pedido de nuevo	**aprobado**
55	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario hace un comentario y lo guarda como negativo	el sistema guarda el comentario para ese negocio y no permite comentar ese pedido de nuevo	**aprobado**
56	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario carga sus datos personales guarda los cambios	el sistema guarda la información del usuario	**aprobado**
57	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario edita sus datos personales guarda los cambios	el sistema guarda la información del usuario	**aprobado**
58	C.9 Perfil/ Resumen de cuenta	Cliente final	el usuario ingresa un campo no válido y guarda los cambios	el sistema indica que el campo no es válido y no permite guardar	**no aprobado**
59	P.1 Inicio de sesión de proveedor	Proveedor	el usuario intenta iniciar sesión con un usuario y contraseña válidos	se inicia la sesión correctamente y se redirige al usuario a la pantalla principal	**aprobado**
60	P.1 Inicio de sesión de proveedor	Proveedor	el usuario intenta iniciar sesión con un usuario válido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
61	P.1 Inicio de sesión de proveedor	Proveedor	el usuario intenta iniciar sesión con un usuario inválido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
62	P.1 Inicio de sesión de proveedor	Proveedor	el usuario intenta iniciar sesión con un usuario válido y sin contraseña	el sistema no permite iniciar sesión e indica que el usuario o	**aprobado**

				contraseña son inválidos	
63	P.2.1 Marcar disponibilidad	Proveedor	el usuario modifica la disponibilidad de un producto y guarda los cambios	el sistema guarda los cambios y notifica al usuario	**no aprobado**
64	P.2.1 Marcar disponibilidad	Proveedor	el usuario modifica la disponibilidad de un producto y no guarda los cambios	el sistema no guarda los cambios	**aprobado**
65	P.2.2 Modificar precio	Proveedor	el usuario modifica el precio de un producto y guarda los cambios	el sistema guarda los cambios y notifica al usuario	**no aprobado**
66	P.2.2 Modificar precio	Proveedor	el usuario modifica el precio de un producto y no guarda los cambios	el sistema no guarda los cambios	**aprobado**
67	P.2.3 Alta de producto	Proveedor	el usuario da de alta un producto válido	el sistema crea el nuevo producto	**aprobado**
68	P.2.3 Alta de producto	Proveedor	el usuario da de alta una promoción válida	el sistema crea la nueva promoción	**aprobado**
69	P.2.3 Alta de producto	Proveedor	el usuario da de alta un producto con datos insuficientes	el sistema indica que faltan datos y no da de alta el producto	**aprobado**
70	P.3.1 Tomar pedido	Proveedor	el usuario acepta un pedido de la lista	el pedido pasa a estado aceptado	**aprobado**

71	P.3.1 Tomar pedido	Proveedor	el usuario acepta un pedido que justo se ha cancelado	el sistema informa que no puede aceptar el pedido	**aprobado**
72	P.3.1.1 Marcar pedido enviado	Proveedor	el usuario marca un pedido de la lista como enviado	el pedido pasa a la lista "en camino"	**aprobado**
73	P.3.2 Rechazar pedido	Proveedor	el usuario rechaza un pedido de la lista	el pedido se rechaza y pasa a la lista de rechazados	**aprobado**
74	P.5 Solicitar atención	Proveedor	el usuario realiza un pedido de atención sin mensaje	el sistema informa que debe ingresar un mensaje para solicitar atención	**aprobado**
75	P.5 Solicitar atención	Proveedor	el usuario realiza un pedido de atención válido	el mensaje se guarda correctamente	**aprobado**
76	P.6 Modificar costo de envío	Proveedor	el usuario confirma el costo sin ningún valor	el sistema muestra un mensaje de error y no permite guardar el valor	**aprobado**
77	P.6 Modificar costo de envío	Proveedor	el usuario confirma el costo con un número negativo	el sistema muestra un mensaje de error y no permite guardar el valor	**no aprobado**
78	P.6 Modificar costo de envío	Proveedor	el usuario confirma el costo con un numero valido	el sistema guarda el valor ingresado	**aprobado**
79	A.1 inicio de sesión de usuario	Administrador	el usuario intenta iniciar sesión con un usuario y contraseña válidos	se inicia la sesión correctamente y se redirige al usuario a la pantalla principal	**aprobado**
80	A.1 inicio de sesión de usuario	Administrador	el usuario intenta iniciar sesión con un usuario válido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**

81	A.1 inicio de sesión de usuario	Administrador	el usuario intenta iniciar sesión con un usuario inválido y una contraseña inválida	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
82	A.1 inicio de sesión de usuario	Administrador	el usuario intenta iniciar sesión con un usuario válido y sin contraseña	el sistema no permite iniciar sesión e indica que el usuario o contraseña son inválidos	**aprobado**
83	A.3.1 ABM Proveedor:	Administrador	el usuario da de alta un proveedor con todos los campos completos y válidos	el sistema da de alta el proveedor	**aprobado**
84	A.3.1 ABM Proveedor:	Administrador	el usuario da de alta un proveedor con campos incompletos	el sistema informa que hay campos incompletos y no permite guardar	**aprobado**
85	A.3.1 ABM Proveedor:	Administrador	el usuario da de alta un proveedor con campos inválidos	el sistema informa que hay campos inválidos y no permite guardar	**aprobado**
86	A.3.1 ABM Proveedor:	Administrador	el usuario elimina un proveedor	el sistema elimina el proveedor seleccionado	**aprobado**
87	A.3.1 ABM Proveedor:	Administrador	el usuario modifica un dato de un proveedor	el dato modificado se guarda correctamente	**aprobado**
88	A.4.1 ABM Negocio:	Administrador	el usuario da de alta un negocio con todos los campos completos y válidos	el sistema da de alta el negocio	**aprobado**
89	A.4.1 ABM Negocio:	Administrador	el usuario da de alta un negocio con campos incompletos	el sistema informa que hay campos incompletos y no permite guardar	**aprobado**

90	A.4.1 ABM Negocio:	Administrador	el usuario da de alta un negocio con campos inválidos	el sistema informa que hay campos inválidos y no permite guardar	**aprobado**
91	A.4.1 ABM Negocio:	Administrador	el usuario elimina un negocio	el sistema elimina el negocio seleccionado	**aprobado**
92	A.4.1 ABM Negocio:	Administrador	el usuario modifica un dato de un negocio	el dato modificado se guarda correctamente	**aprobado**
93	A.4.2 Ver pedidos de atención	Administrador	el usuario hace click en ver detalle de un pedido de atención	el sistema muestra el detalle del pedido de atención	**aprobado**
94	A.4.2 Ver pedidos de atención	Administrador	el usuario marca un pedido como en proceso	el pedido pasa a estado "en proceso"	**aprobado**
95	A.4.2 Ver pedidos de atención	Administrador	el usuario marca un pedido como solucionado	el pedido pasa a estado "solucionado"	**aprobado**
96	A.5 Cobranza	Administrador	el usuario accede a los datos de un periodo calculado	el sistema muestra por negocio el total a pagar y el total general	**aprobado**
97	A.5 Cobranza	Administrador	el usuario accede a los datos de un periodo sin calcular	el sistema no muestra ningún dato	**aprobado**

Tabla 10-1 Casos de prueba y resultados de las pruebas funcionales

La siguiente tabla muestra el seguimiento realizado a los casos no aprobados:

Caso	Comentario	Solución
16	el sistema no permite continuar pero no da ningún aviso de que no encontró la dirección	se agrega un aviso de que la dirección no se pudo resolver
28	el mensaje de error persiste al elegir otro producto	se limpia el mensaje al abrir la ventana
58	el formulario no valida la cantidad máxima de caracteres	se agrega la validación
64	los cambios se guardan pero no se muestra ningún mensaje	se agrega mensaje con el resultado
66	los cambios se guardan pero no se muestra ningún mensaje	se agrega mensaje con el resultado
78	el sistema no valida el valor negativo	se agrega la validación

Tabla 10-2 Soluciones casos no aprobados

Resumen de las pruebas realizadas:

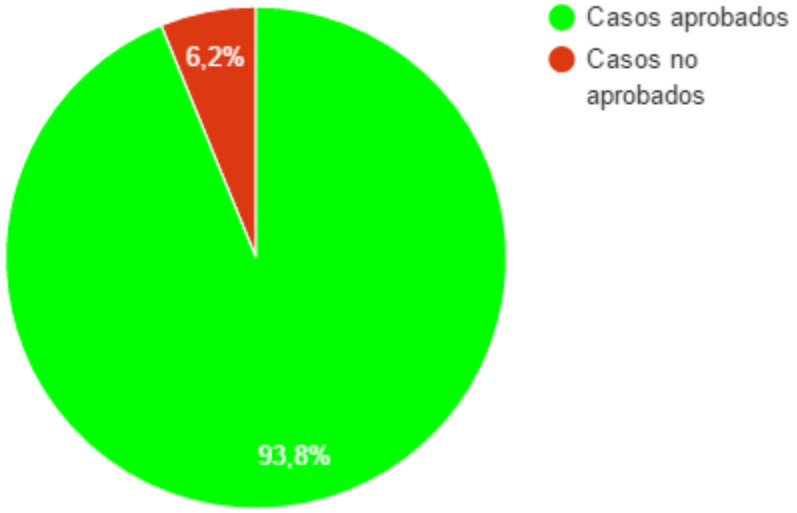

Figura 10-8 Gráfico pruebas realizadas

c. Pruebas de performance con Gatling

¿Cómo trabaja específicamente Gatling?

Para cada entidad generada en la aplicación crea una simulación, la cual accede a los servicios definidos en el controlador de la misma y realiza las pruebas sobre los mismos. También cuenta con un archivo de configuración donde se puede configurar el número de usuarios autenticados y de solicitudes a realizar. En la siguiente imagen se muestra el comienzo de la ejecución (con un comando maven) y la solicitud de ingreso del número de simulación:

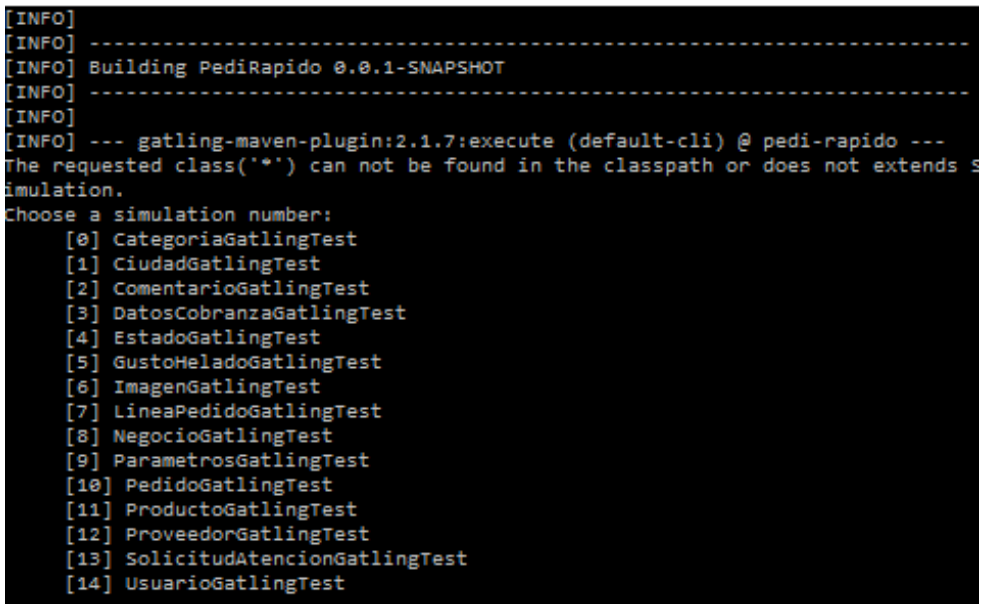

Figura 10-9 Simulaciones Gatling

Luego, una vez que se da comienzo la simulación, se puede observar por consola las pruebas parciales realizadas y los resultados de las mismas.

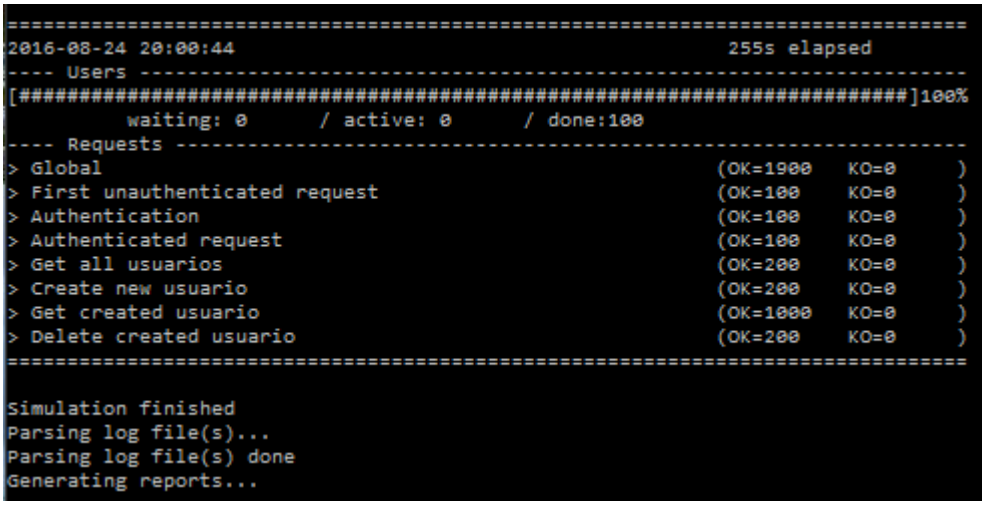

Figura 10-10 Corriendo prueba Gatling

Enfocándonos en nuestro proyecto, realizamos las simulaciones sobre entidades que consideramos críticas para el core de nuestro negocio. Decidimos realizarla sobre la entidad Negocio debido a que tiene alto impacto sobre los Usuarios ya que constantemente solicitan recuperar todos los negocios para elegir donde realizar un pedido y sobre la entidad Pedido, que tiene alto impacto en los Proveedores, ya que constantemente requieren recuperar todos los pedidos asociados a su negocio.

A continuación, mostramos gráficos generados en los resultados de la corrida de la simulación. La idea es comparar los mismos entre las dos entidades y sacar conclusiones de ello.

Entidad Negocio.

En este primer gráfico, podemos ver una tabla donde se muestran los request a los que la simulación accedió (obtener todos los negocios, crear un negocio, eliminar un negocio). En las 5 columnas pertenecientes a “Executions” se puede ver la cantidad de request realizados y la cantidad de ellos que fueron exitosos (“OK”) y no exitosos (“KO”), para esta simulación todos fueron exitosos, es por eso que para todos os request el %KO es 0%. Luego, el resto de la tabla muestra los valores en milisegundos (ms) del tiempo de respuesta.

STATISTICS — Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	1900	1900	0	0%	7.546	3	5	5	100	115	142	10	23
First un... request	100	100	0	0%	0.397	3	5	5	15	16	17	6	3
Authentication	100	100	0	0%	0.397	96	105	109	134	139	142	109	10
Authenti... request	100	100	0	0%	0.397	5	6	6	14	15	19	6	2
Get all negocios	200	200	0	0%	0.794	3	5	6	12	16	26	5	2
Create new negocio	200	200	0	0%	0.794	3	4	4	9	16	100	4	7
Get created negocio	1000	1000	0	0%	3.971	4	5	5	11	13	76	5	3
Delete c... negocio	200	200	0	0%	0.794	3	4	4	9	10	20	4	2

Figura 10-11 Resultado prueba con Gatling – 1

Con respecto a los tiempos de respuesta, podemos ver que el tiempo medio más alto es el de la autenticación (109 ms). Pero queremos apuntar a los siguientes request:

- **Obtener todos los negocios:** con un tiempo de respuesta medio de 5 ms, un desvío estándar de 2 ms, un tiempo mínimo y máximo de 3 ms y 26 ms respectivamente.
- **Crear un negocio nuevo:** con un tiempo de respuesta medio de 4 ms, un desvío estándar de 7 ms, un tiempo mínimo y máximo de 3 ms y 100 ms respectivamente.

Específicamente para las solicitudes de recuperar todos los negocios:

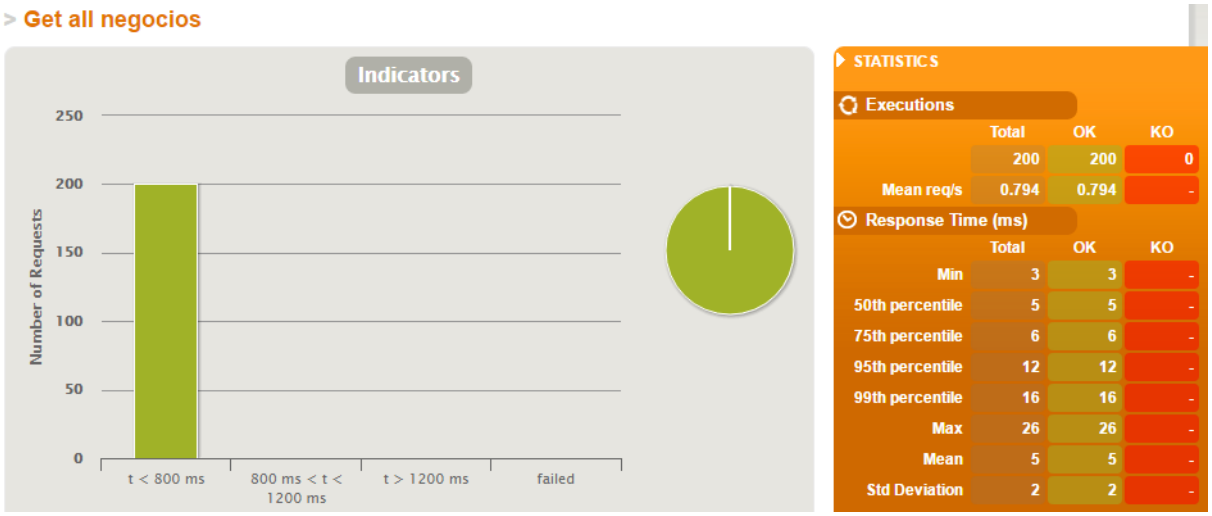

Figura 10-12 Resultado prueba con Gatling - 2

Luego, en las siguientes dos imágenes se puede ver el número de solicitudes y el número de respuestas por segundo durante un lapso de tiempo (4 minutos aproximadamente), en relación también con la cantidad de usuarios activos.

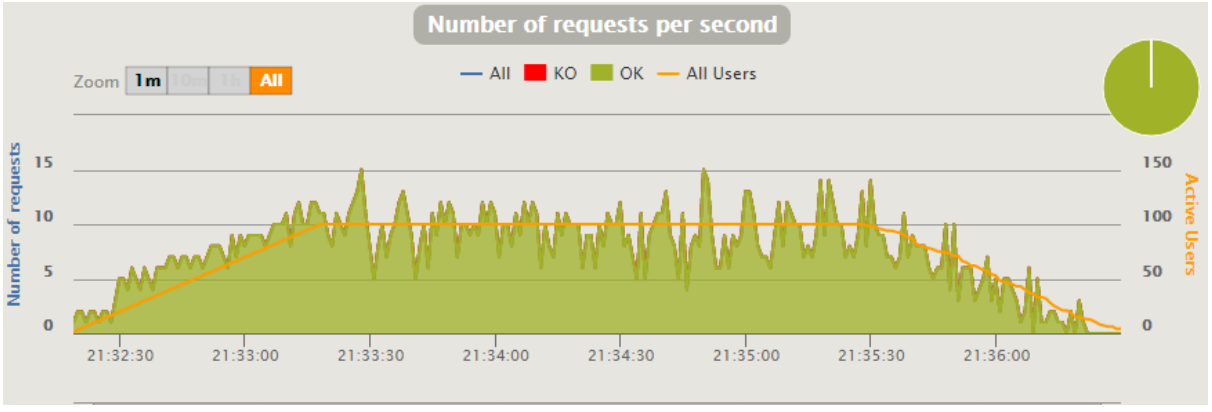

Figura 10-13 Resultado prueba con Gatling - 3

Para el caso del número de solicitudes por segundo, podemos ver que todas fueron exitosas. También que el valor más alto registrado fue de 15 solicitudes (siendo alcanzado 2 veces, por lo menos). Además, que el número máximo de usuarios activos fue de 100 usuarios.

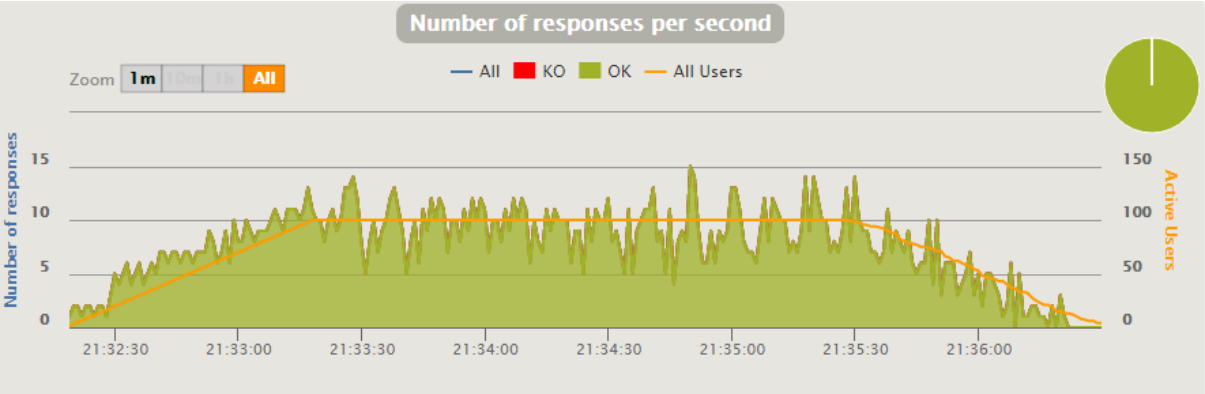

Figura 10-14 Resultado prueba con Gatling - 4

Ahora, para el caso del número de respuestas por segundos también el número máximo de usuarios activos fue de 100. Siendo también de 15 respuestas por segundos el valor más alto, pero registrado en una sola oportunidad.

Entidad Pedido.

Ya explicamos en la entidad Negocio la tabla con los resultados de la simulación. Observando los resultados de la simulación de la entidad Pedido podemos ver que todos los request realizados fueron exitosos, siendo de 0% el porcentaje de los no exitosos (“KO”).

STATISTICS — Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	1900	1900	0	0%	7.344	3	6	10	111	142	1029	17	52
First un... request	100	100	0	0%	0.387	4	6	7	24	279	663	18	70
Authentication	100	100	0	0%	0.387	109	116	128	150	1029	1029	147	143
Authenti... request	100	100	0	0%	0.387	6	9	13	42	371	377	28	75
Get all pedidos	200	200	0	0%	0.773	5	7	10	16	118	119	10	17
Create new pedido	200	200	0	0%	0.773	3	5	7	10	62	300	7	21
Get created pedido	1000	1000	0	0%	3.865	4	5	9	14	28	134	7	6
Delete c...d pedido	200	200	0	0%	0.773	3	5	9	24	38	46	8	7

Figura 10-15 Resultado prueba con Gatling - 5

Con respecto a los tiempos de respuesta, podemos ver que el tiempo medio más alto es el de la autenticación, al igual que con la entidad Negocio, siendo de 147 ms. Pero queremos apuntar a los siguientes request:

- **Obtener todos los pedidos:** con un tiempo de respuesta medio de 10 ms, un desvío estándar de 17 ms, un tiempo mínimo y máximo de 5 ms y 119 ms respectivamente.
- **Crear un pedido nuevo:** con un tiempo de respuesta medio de 7 ms, un desvío estándar de 21 ms, un tiempo mínimo y máximo de 3 ms y 300 ms respectivamente.

Específicamente para las solicitudes de recuperar todos los pedidos:

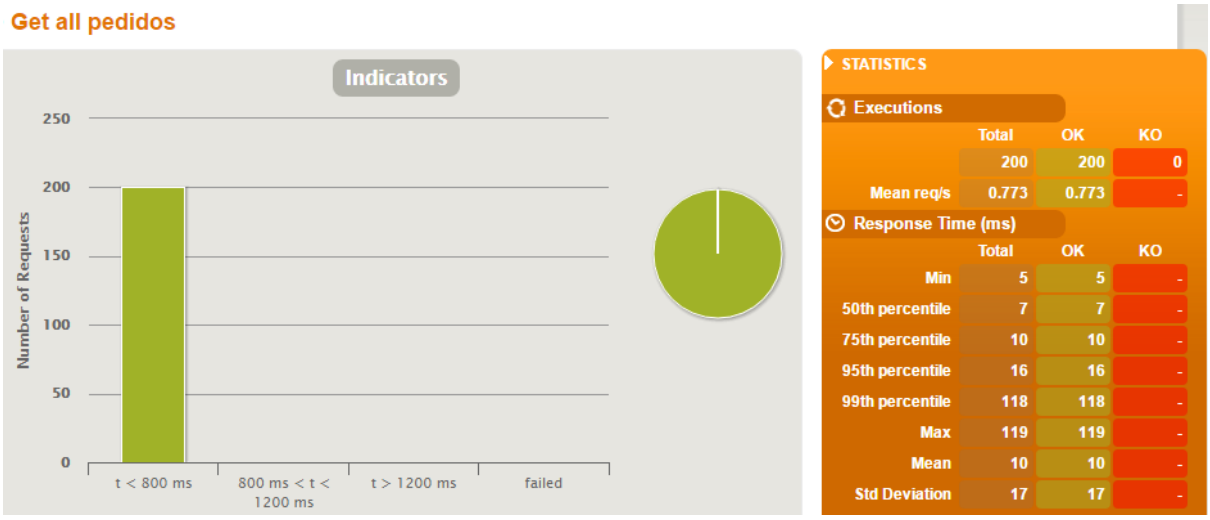

Figura 10-16 Resultado prueba con Gatling - 6

Luego, en las siguientes dos imágenes se puede ver el número de solicitudes y el número de respuestas por segundo durante un lapso de tiempo (también de 4 minutos aproximadamente), en relación con la cantidad de usuarios activos.

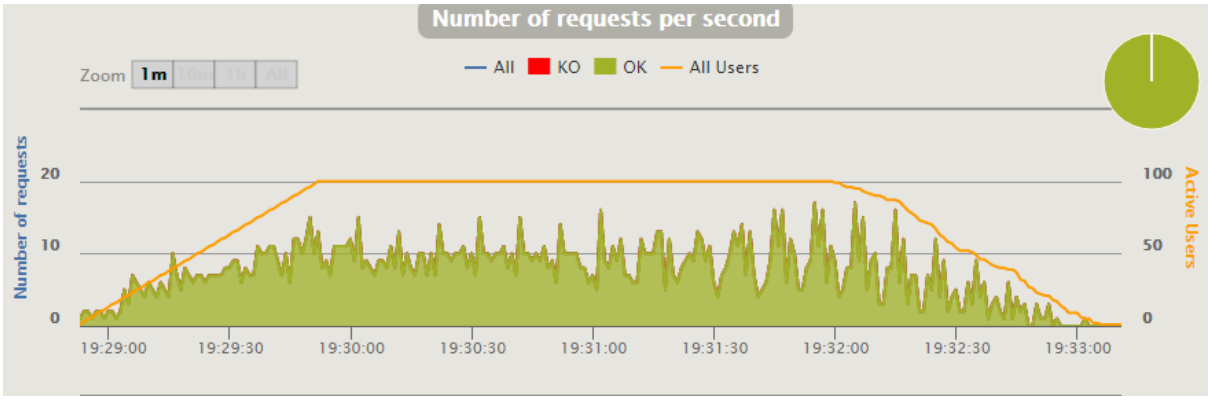

Figura 10-17 Resultado prueba con Gatling - 7

Para el caso del número de solicitudes por segundo, podemos ver que todas fueron exitosas. El número máximo de usuarios activos fue de 100 usuarios y también que el valor más alto superó los 15 ms en por lo menos 5 momentos.

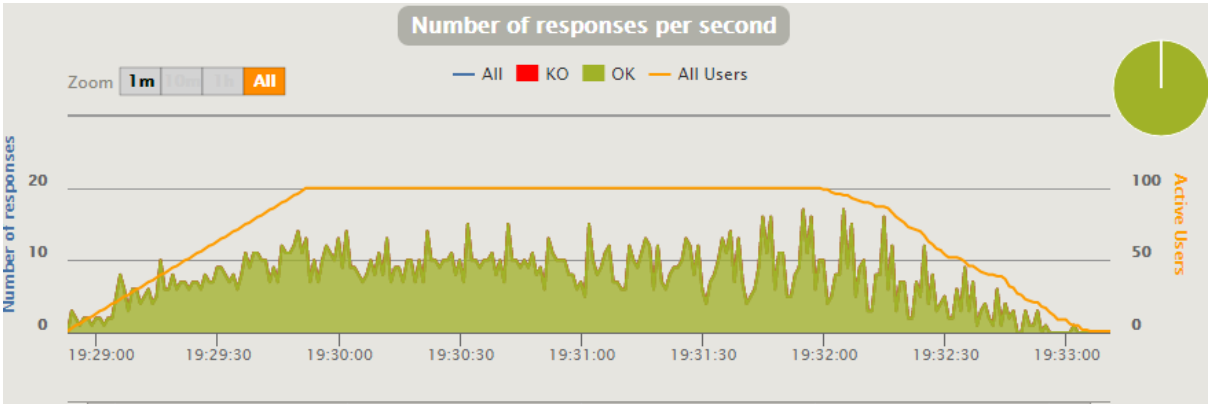

Figura 10-18 Resultado prueba con Gatling - 8

Ahora, para el caso del número de respuestas por segundos también el número máximo de usuarios activos fue de 100. Siendo superado los 15 ms el tiempo de respuestas por en, al menos, 4 oportunidades que dan respuesta a las solicitudes realizadas en los mismos períodos de tiempo.

Conclusiones de las pruebas:

Con la configuración inicial que propusimos intentamos imitar las condiciones reales de producción de la aplicación, ya sea en la concurrencia de usuarios y también con las solicitudes que más se realizan desde los distintos roles. Los resultados obtenidos con esta herramienta fueron satisfactorios, para el informe decidimos mostrar las simulaciones de dos entidades que considerábamos críticas, pero en verdad realizamos simulaciones de varias entidades y todas dieron muy buenos resultados.