Universidad ORT Uruguay
Facultad de Ingeniería

Random House App: Plataforma integral para la gestión de bares

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

Juan Santiago Drets Pons - 209326
Rodrigo Duarte Katcher - 217374
Sebastián Gustavo Rodríguez Janeiro - 192412

Tutor: Marcelo Cagnani

2021

Declaración de autoría

Nosotros, Juan Drets, Rodrigo Duarte y Sebastián Rodríguez, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizamos el Proyecto Final de Ingeniería en Sistemas;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente que fue construido por otros, y que fue construido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado aclaraciones correspondientes.

Juan Drets	Rodrigo Duarte	Sebastián Rodríguez
18/09/2021	18/09/2021	18/09/2021

Agradecimientos

Nos gustaría agradecer a diferentes personas que de alguna manera formaron parte del proceso.

En primer lugar, a nuestros familiares y amigos los cuales han estado presentes brindando un apoyo incondicional no solo a lo largo de este proyecto sino también durante toda la carrera.

Además, nos gustaría agradecer a nuestro tutor, Marcelo Cagnani que nos guio desde el principio del proyecto hasta el final y fueron fundamentales todos sus aportes para llegar al mejor resultado posible.

Por otro lado, también queremos agradecer a los revisores Leonardo Scafarelli, Rafael Bentancour y Amalia Álvarez, quienes nos ofrecieron valioso *feedback* luego de cada revisión con el fin de mejorar nuestro trabajo. También fueron muy importantes los expertos con los cuales tuvimos reuniones para mejorar áreas específicas, estos fueron Gastón Mousques, Martin Solari y Amalia Álvarez.

Por último, un especial agradecimiento a nuestro cliente Random House Srl con quienes en conjunto pudimos construir el proyecto.

Agradecemos a todos los mencionados anteriormente.

Abstract

Random House App es un sistema integral de toma y gestión de comandas para un bar cervecero que busca transformar a través de la tecnología su modelo de atención. Surge debido a su necesidad actual, la cual presenta problemas en el flujo y control de los pedidos.

El proyecto busca generar una plataforma multipropósito que le permita al bar acelerar el proceso de atención, ordenar y controlar la gestión de comandas, llevar un manejo del stock, obtener estadísticas de ventas visualizadas en *dashboards* mejorando la toma de decisiones y generar fidelidad con sus clientes.
Dicho sistema cuenta con las siguientes soluciones para los distintos usuarios, estas son:

- Terminal de autoservicio en donde los clientes realizan los pedidos.
- Aplicación móvil para que los usuarios realicen pedidos desde sus celulares pudiendo acceder al historial y programas de fidelización.
- Web de toma de comandas para los pedidos en caja.
- Terminal de gestión de comandas para la cocina.
- Backoffice para gestionar los pedidos y la información referente al bar en conjunto.

Para cumplir con la correcta realización del sistema se utilizó un ciclo de vida iterativo e incremental el cual se adapta perfectamente a la situación del proyecto, se inicia con un mínimo producto viable para continuar iterando hasta construir la solución más adecuada. Para su óptima gestión se sigue un enfoque ágil que permite trabajar iterativamente agregando cambios y mejoras de manera rápida y dinámica.
Durante el proceso el equipo utilizó un conjunto de métodos, herramientas y técnicas para asegurar la calidad del proyecto donde se controló que cada aspecto sea llevado a cabo de la mejor manera. Se recolectaron distintas métricas no solo para medir los resultados obtenidos, sino también para poder mejorar los procesos de gestión y desarrollo.
De esta manera se concluyó en un sistema completo que logró satisfacer al cliente y ser puesto en producción.

Palabras clave

Random House; Bar; Comandas; Cocina; Autoservicio; Clientes; Terminales; Mesas; Ruby; Rails; Flutter; Metodologías ágiles; Scrum; Proyecto; Sistema; Software; Universidad; ORT;

Glosario

Addons: Componente de software el cual hace que mejore las capacidades o el rendimiento informático. [1]

API: Conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones. Permite que productos y servicios se comuniquen unos con otros sin la necesidad de saber cómo están implementados [2].

Backend: Es la parte del desarrollo web que se encarga de que toda la lógica de un sistema funcione. Son un conjunto de acciones que son invisibles para los usuarios, por ejemplo, la comunicación con el servidor. [3]

Beeper: Dispositivo de telecomunicaciones que recibe mensajes cortos, también llamado dispositivo buscapersonas. [4]

Bug: Error de código en un programa informático. [5]

Burndown Chart: Gráfica que representa la cantidad de trabajo realizado en un *sprint*. En el eje “x” se representa el tiempo, mientras que en eje “y” la cantidad de story points restantes. [6]

Code review: Examen sistemático que se realiza sobre el código fuente de un programa informático. Es una técnica utilizada para mejorar las cualidades de los desarrolladores involucrados mediante la discusión de posibles mejoras en el código. [7]

Daily meeting: Reunión diaria del equipo, sirve para que todos los miembros del mismo estén sincronizados con las tareas a realizar. [8]

Dashboard: Herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los indicadores clave de desempeño, métricas y datos fundamentales para hacer un seguimiento del estado de una empresa, un departamento o un proceso específico. [9]

Feature: Unidad funcional de un sistema de software que satisface un requisito. [10]

Feedback: Es la acción de ofrecer información a una persona sobre un resultado. [11]

Firebase: plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida. [12]

Flutter: Framework creado por Google que tiene como finalidad el crear interfaces de software. [13]

Frontend: Parte del desarrollo web que se dedica al diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos. [14]

Framework: Estructura base utilizada como punto de partida para elaborar un proyecto con objetivos específicos. [15]

Full Stack: Programador con un perfil técnico muy completo que conoce bien tanto lo referente a back-end como lo referente a front-end. [16]

Git: Software de control de versiones. [17]

HTTPS: Es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio web. [18]

Ingeniería inversa: Proceso que identifica las propiedades de un objeto físico mediante la realización de un análisis exhaustivo de su estructura, funciones y operaciones. [19]

JSON: Formato de texto sencillo para el intercambio de datos. [20]

Jira: Software diseñado para ayudar a equipos de todo tipo a gestionar el trabajo. [21]

Material Design: Es un sistema de diseño creado por Google para ayudar a equipos a construir experiencias digitales de alta calidad para Android, iOS, Flutter y la web. [22]

Mock Ups: Fotomontajes que permiten a los diseñadores gráficos y web mostrar al cliente como quedarán sus diseños. [23]

MVP: Un producto mínimo viable es un producto de software que cuenta con las características, requerimientos y requisitos mínimos para que un software pueda ser lanzado al mercado. [24]

Planning Poker: Técnica para calcular una estimación basada en el consenso, en su mayoría utilizada para estimar el esfuerzo o el tamaño relativo de las tareas de desarrollo de software. [25]

Product Owner: Encargado de maximizar el valor de una solución generada en las empresas a partir de las metodologías ágiles. [26]

Product Backlog: Listado ordenado y priorizado de los requisitos necesarios para la implementación de un proyecto. [27]

Pull Request: Acción de validar un código que se va a sumergir de una rama a otra. Es un proceso de validación. [28]

Push notifications: Son mensajes que se envían de forma directa desde el servidor a dispositivos móviles para interactuar con los usuarios. [29]

Ruby: Lenguaje de programación de propósito general. [30]

RubyGems: Gestor de paquetes para el lenguaje de programación Ruby que proporciona un formato estándar y auto contenido (llamado gem) para poder distribuir programas o bibliotecas en Ruby. [31]

Rails: Framework de desarrollo de aplicaciones web escrito en el lenguaje de programación Ruby. [32]

Release: Versión que se distribuye a los clientes. [33]

REST: Interfaz para conectar varios sistemas basados en el protocolo HTTP. [34]

SCM: Refiere a la optimización de la creación y el flujo de un producto a lo largo de todo el proceso del mismo. [35]

Scrum: Metodología ágil que consiste en trabajar en equipo a través de iteraciones o *Sprints*.

Sprint: Dentro de Scrum, un *sprint* es un intervalo prefijado de tiempo durante el cual se crea un incremento de producto utilizable, potencialmente entregable.

Story Points: Unidad de medida para expresar o estimar el esfuerzo general que sería necesario para fabricar un elemento de un producto. Son utilizados por equipos de Scrum.

Índice

1. Introducción

1.1. Descripción del proyecto

Random House es un proyecto que fue realizado como requisito para la obtención del título “Ingeniero en sistemas” de la Universidad ORT Uruguay para los estudiantes Juan Drets, Rodrigo Duarte y Sebastián Rodríguez. Además, también como solución para nuestro cliente Random House SRL.

Dicho proyecto es una plataforma integral de gestión y toma de pedidos que pretende mejorar la experiencia de los usuarios del bar como también mejorar la gestión de pedidos mediante la automatización de procesos. El sistema está compuesto por una aplicación para terminales autoservicio de toma de pedidos, una aplicación híbrida (iOS y Android) para la toma de pedidos, una web para que los cajeros puedan tomar los pedidos presenciales, un sistema de recepción y gestión de comandas para la cocina y un backoffice web en donde los administradores del bar pueden gestionar la información de la carta como también las comandas y otras funcionalidades del sistema.

1.2. Objetivos

Como parte del proyecto se decidió fijar objetivos para poder tener metas a las cuales se quería llegar para luego medir y decidir si se concretaron exitosamente. De esta manera el equipo podía tener presente en todo momento a que se apuntaba llegar. Para esto se dividió en las tres categorías siguientes:

1.2.1. Objetivos académicos

- Poner en práctica todo lo aprendido en la carrera

Como principal objetivo académico se encontró que este es muy importante, ya que el equipo considero que el proyecto de fin de carrera es la instancia en la cual como estudiantes se tiene que aprovechar y reunir todo lo aprendido para volcarlo en un mismo proyecto.

- Lograr la excelencia académica

Se planteó como objetivo muy importante poder lograr la excelencia académica siendo esta de un 97% o superior ya que se cree que si se utiliza lo aprendido en la carrera se puede obtener un excelente resultado que derivaría en la excelencia académica.

- Aprender nuevas tecnologías

En conjunto con los anteriores objetivos resultó importante el hecho de aprender al menos dos nuevas tecnologías ya que supone un desafío para los integrantes y de valor para crecimiento profesional.

1.2.2. Objetivos del equipo

- Aprender a trabajar en equipo aplicando metodologías de trabajo

Los objetivos del equipo van en consonancia con los objetivos académicos ya que se planteó trabajar como un equipo y a la vez usar metodologías de trabajo. Lo que implicaría utilizar en conjunto la teoría de las metodologías con las habilidades blandas. Para evaluar si se trabajó correctamente se realiza una encuesta de satisfacción de trabajo del equipo.

- Aprender a tratar con clientes reales

Este objetivo es importante ya que por la naturaleza de la tesis se tiene la posibilidad de trabajar con un cliente real con una necesidad real. Lo que implica que comprender los requerimientos del cliente sea crucial para el desarrollo del proyecto. Para evaluar el trabajo con el cliente se realiza una encuesta de satisfacción.

1.2.3. Objetivos del producto

- Crear un producto de valor para el cliente generando impacto en su negocio

Es clave que el proyecto logre aportar valor al cliente y este logre quedar satisfecho. Al igual que en el objetivo anterior este se evalúa en la encuesta de satisfacción del cliente.

- Ser innovadores en el segmento del mercado

Un factor muy importante en estos tipos de proyecto es la innovación ya que al cliente le sirve no solo para automatizar procesos sino también para tener más repercusiones en el rubro.

- Crear un sistema intuitivo para los clientes del bar

Como el sistema será utilizado por clientes de todo tipo, crear un sistema intuitivo es central en este tipo de proyectos. Ya que una buena experiencia invita a los usuarios a que repitan su uso y que queden satisfechos con el bar.

2. Problema actual

2.1. Nuestro cliente

Random house SRL es una cervecería de casi dos años de antigüedad localizada en el barrio Parque Rodó, Montevideo, Uruguay. La misma fue fundada por un grupo de socias y se caracteriza por ser innovador, tener un público joven y tener juegos para sus clientes. Se encuentra abierto de jueves a sábado de 19:30 a 02:00hs. El rango de edad de los clientes suele ser entre los 23 y 29 años. Por mes reciben un promedio de 1300 personas. El modelo de negocio está basado en la venta de cerveza artesanal (incluyendo su propia línea) y comida rápida.

La relación con el cliente surge ya que uno de los integrantes del equipo conocía a las dueñas las cuales se encontraban buscando cómo innovar en el bar y optimizar el flujo de pedidos. Como el equipo se encontraba próximo a comenzar la tesis se le propuso la idea para ver si existía la posibilidad de ser realizada.

2.2. Contexto

Para explicar el contexto es necesario aclarar que se consideraron dos escenarios distintos, el actual (en pandemia [36]) y el anterior (sin pandemia). Esto es importante ya que estos dos contextos son diferentes y han modificado la dinámica de trabajo y de atención del bar.

Esta distinción se hace por lo antedicho y porque el cliente piensa volver a la vieja modalidad de trabajo.

Comandas

A continuación, se detallan ambas modalidades de trabajo.

En el contexto pre pandemia el flujo consiste en: el cliente llega al bar, se dirige a la barra/caja en donde puede ver la carta y una vez que decide lo que va a pedir, lo hace directamente a la persona de caja, la cual imprime la comanda del pedido para mandarlo a cocina y luego le entrega un *beeper* al usuario que esperará en su mesa a que el pedido esté listo. Una vez que lo está, se hace sonar el *beeper* para que el

cliente prosiga a retirar su pedido. En el caso de haber pedido bebida se lo indica al empleado de la barra y éste se lo entrega al usuario en el momento. Si se quiere pedir comida y/o bebida nuevamente se repite el proceso anterior. En todos los casos el cliente deberá abonar lo pedido en el momento de la compra.

En el contexto actual, es decir en pandemia, el flujo consiste en: el cliente llega al bar y se dirige a una mesa, una vez acomodado debe escanear un código QR que se encuentra en la misma donde puede acceder al menú del bar. Una vez que decide lo que va a consumir debe agregar como contacto el número de Random House que se encuentra en la carta y deberá hablarle por WhatsApp [37] para realizar su pedido e indicar en qué mesa se encuentra sentado. La persona que se encuentra en la caja es la que se encarga de responder los mensajes vía WhatsApp e imprimir las comandas para llevarlas a cocina. Una vez que realiza su pedido deberá esperar en la mesa que se encuentra, y cuando el mismo esté finalizado el mozo se lo llevará. Al cliente querer retirarse del local, se lo indica al mozo y éste le cobrará el monto correspondiente (previamente corroborando el pedido hecho por WhatsApp).

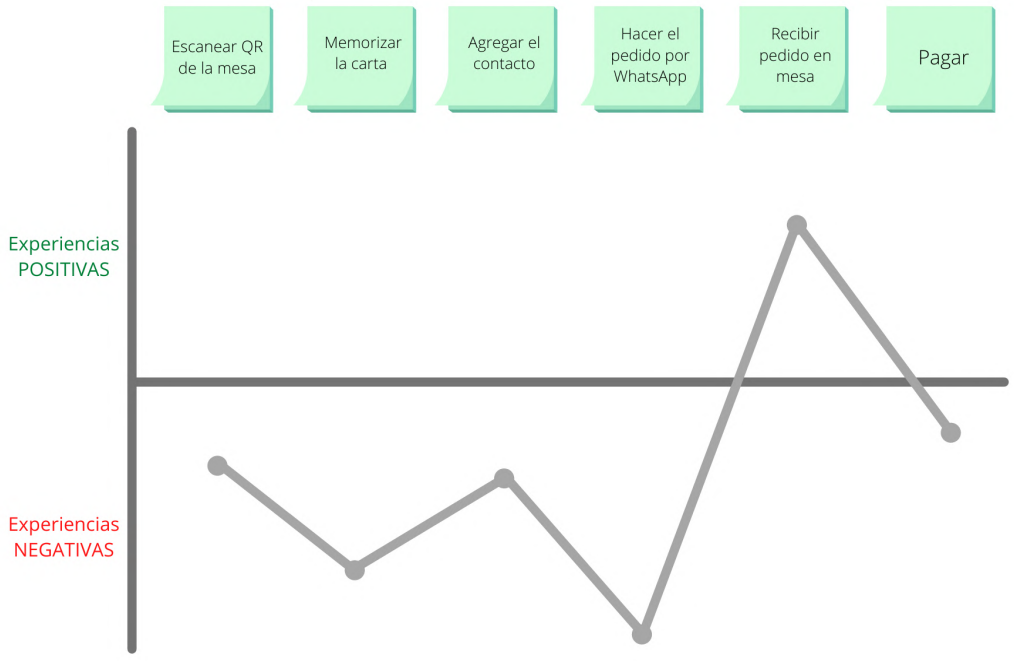


Figura 1: Customer Journey

Control de stock

Actualmente el control de *stock* se realiza haciendo un conteo de lo consumido cada 15 días anotando en una libreta. El cliente indicó que suele ser poco práctico y poco preciso a la hora de llevar un registro.

Software de comandas

Actualmente nuestro cliente cuenta con un *software* de comandas (DelySoft [38]) el cual es utilizado solamente para llevar el registro de los pedidos de cada mesa.

Cobros

Actualmente el bar acepta efectivo y pago con tarjeta.

2.3. El problema

Tanto en la situación actual como pre covid, se le presentan los siguientes problemas o dolores a nuestro cliente y a sus usuarios.

- Lenta realización de pedidos

En el contexto pre pandemia los usuarios cada vez que querían realizar un nuevo consumo se debían que acercar a la caja para hacer el pedido y luego nuevamente para hacerse del mismo, lo que implica una acción costosa para los usuarios ya que tienen que dejar de hacer lo que están haciendo para consumir más, esto puede derivar en un menor consumo de los usuarios en Random House.

En los horarios pico este problema se acrecienta ya que se forman filas para realizar los pedidos y esto aumenta los tiempos de espera de los usuarios afectando directamente la experiencia.

- Realización del pedido dificultosa

En el contexto actual, el costo de realizar un nuevo pedido para el cliente es muy alto ya que no solo tiene que acceder a la carta escaneando un código QR, sino que también debe memorizar la carta, agregar al bar como nuevo contacto y enviarle su pedido.

- Realización del pedido costosa

Del lado de Random House deben tener un empleado constantemente pendiente de los mensajes para imprimir las comandas y enviarlas a cocina. Por lo que necesitan una gran cantidad de empleados para atender correctamente a sus clientes. Dependiendo el día es la cantidad de empleados que tienen, generalmente tienen entre 5 y 7 empleados trabajando en el flujo de toma y creación de comandas.

- Falta de control

En adición a los problemas anteriores, se tiene poca información sobre los clientes, lo que genera que muchas de las decisiones que se toman no sean basadas en base a datos. Además, tampoco se tiene un sistema definido de control de stock.

- Desorden en las comandas

El hecho de que las comandas sean realizadas a mano e impresas deriva en que se genere desorden en la cocina sobre qué comandas llegaron antes y cuáles después. También complejiza y genera pérdida de tiempo a la hora de realizar cambios sobre las mismas.

- Falta de fidelización con los clientes

En el bar no se tiene un sistema definido de fidelización de clientes. Hoy en día el único medio de comunicación es a través de Instagram.

2.4. Desafíos del proyecto

Este proyecto supuso una serie de desafíos que el equipo ha tenido que sobreponer en el transcurso de su desarrollo. No solo los que fueron presentados explícitamente por nuestro cliente sino también los encontrados por el equipo en el *sprint* 0 (el cual se detalla en la sección 4.3. Metodología de trabajo) y los propios de un proyecto de esta naturaleza.

Dentro de los desafíos decidimos separar en categorías para proceder en la explicación de cada una. Tales categorías son: desafíos de gestión, desafíos del equipo, desafíos tecnológicos, desafíos del producto y desafíos del contexto actual.

2.4.1. Desafíos de gestión

Dentro de los desafíos de gestión el equipo se encontró con que ninguno de sus integrantes había trabajado como responsable de un proyecto, lo que supuso tomar roles que no habían ocupado antes y tener responsabilidades nuevas. Por lo antedicho es que también se presentaron desafíos de cómo gestionar el trabajo y las estimaciones de los tiempos como también la comunicación con el cliente. Debido a la falta de experiencia es que lograr definir el *framework* adecuado de gestión fue un desafío importante ya que tenía que servirle al equipo a lo largo de todo el proyecto. Dentro del *framework* elegido también hubo que decidir qué cosas tomar y cuáles modificar para las necesidades del equipo.

2.4.2. Desafíos del equipo

Dentro de los desafíos del equipo, se encuentran tales como la gestión del tiempo para que cada integrante pueda cumplir con las metas trazadas y poder tener el tiempo suficiente para poder asistir a las reuniones pactadas con el tutor y entre los miembros del equipo.

2.4.3. Desafíos tecnológicos

Dentro del área tecnológica hay una variedad de desafíos, como lo fue la capacitación de tecnologías nuevas para el equipo, también dentro de cada una de esas tecnologías, surgieron desafíos como el poder integrarse con librerías o plugins de terceros o servicios externos, dentro de estos se destacan los presentados a continuación.

Firebase fue necesario integrarlo tanto con el *backend* como el *frontend*, en donde se tuvo que aprender a utilizar las *Firebase Push Notifications*, *Firebase* Authentication [39] y *Firebase* Storage [40].

Para Mercado Pago [41] se tuvo que estudiar y comprender cómo funciona la *API* la cual tiene documentación en constante actualización y no se encuentra muy bien redactada.

Chart Kick [42] fue necesario saber cómo usarlo en el *backend* para la generación de *dashboards*.

Si bien los *deploys* en Heroku [43] y en *Firebase Serve*r son conocidos por no ser muy complicados igualmente requiere comprender su funcionamiento y aprender a realizar los deploys de forma automática en ambos.

Bugsnag [44] es una herramienta para encontrar y registrar problemas no identificados en el sistema ya en funcionamiento. Requiere integración a nivel de código e investigación para aprender a usarla.

2.4.4. Desafíos del producto

Dentro de los desafíos del producto el más importante a criterio del equipo, fue el de lograr comprender íntegramente el producto y sus usuarios. Una parte imprescindible para comprender el producto correctamente era entender al cliente por lo que una óptima comunicación era crucial para el proyecto. Entender el contexto en donde se iba a poner en producción también era muy importante ya que las condiciones no son las normales debido a factores como la iluminación del lugar o el recambio constante de usuarios, lo que desde un principio hizo tener como premisa el lograr una excelente UI/UX [45]. La realización de este proyecto implica la digitalización del proceso de atención a los clientes y de gestión de las comandas por lo que esto supone una gran responsabilidad de parte del equipo ya que se trabaja sobre el flujo principal del negocio. La correcta realización del proyecto deriva en la adaptación y adopción de los clientes al nuevo proceso. Poder entender el alcance del proyecto también fue un desafío importante ya que el objetivo del equipo siempre fue entregar un producto de excelente calidad y que a su vez sea lo más completo posible, una combinación que de ser llevada con rigurosidad es difícil de alcanzar.

2.4.5. Desafíos del contexto actual

El contexto actual, debido a la pandemia, se ha tornado sumamente impredecible, por lo que el equipo debió tener los objetivos claros para poder tener la capacidad de afrontar los desafíos que se interpusieron, de manera de alcanzarlos sin problemas. Debido a que el proyecto comenzó en pandemia, se tuvo presente que

este tema iba a ser desafiante a lo largo del año de trabajo ya sea por cuarentenas de los miembros del equipo, enfermedades o restricciones de movilidad o aforo, siendo esto último muy importante para este proyecto en particular ya que desde un principio se tenía como meta ir en repetidas ocasiones a probar nuestro producto al bar.

2.5. Interesados

Se logró identificar los principales interesados en que el proyecto sea llevado a cabo con éxito, los mismos son los siguientes:

- Socias de Random House, debido a que desde un principio pretendieron poner en producción el resultado generado por el equipo.
- Empleados del bar, ya que se verían afectados positiva y negativamente por nuestro sistema. Negativamente ya que posiblemente la implementación del sistema implique que se realice un recorte de personal ya que se están automatizando procesos, y positivamente ya que se eliminaran tareas tediosas como tener que tomar los pedidos vía WhatsApp y además se eliminan puntos de falla.
- Clientes del bar, ya que se implementará un sistema innovador que mejora la experiencia de realizar pedidos en el bar. Además de tener la posibilidad de acceder a descuentos por la utilización de su aplicación.
- Universidad ORT, ya que como alumnos de la misma la universidad espera un desempeño acorde a lo enseñado en el transcurso de la carrera.
- El equipo, ya que al culminar se cumplen muchos de sus objetivos como son el hecho de obtener el título de Ingeniero en Sistemas y poder generar valor frente a un cliente real.
- Competencia del bar, ya que Random house estaría implementando un sistema que automatiza varios procesos y mejora la experiencia de sus usuarios.

2.6. Solución

Como se mencionó anteriormente, se intentó que el producto fuese lo más completo posible. Dicho esto, se procede con la explicación del mismo y cómo se logró su implementación.

El producto es una plataforma de gestión de un bar que tiene incorporado un sistema multiplataforma de pedidos para sus productos.

Todas las aplicaciones dentro de la solución que fueron desarrolladas han sido pensadas para que se abarque completamente el flujo de pedidos. De esta manera se contemplan los usuarios que desean pedir desde la mesa (aplicación mobile), los que desean pedir desde la terminal de autoservicio, y los que desean pedir directamente en persona desde la caja.

Al contemplar el flujo completo de pedidos se tiene acceso completo sobre las comandas, la información del consumo y movimientos del bar. Esto permite llevar un mejor registro de lo que sucede y de esta manera las gráficas y datos de consumo son muy precisas por lo que se recolecta información de valor para Random House.

Debido a los diferentes roles y sistemas es que para una correcta explicación de la solución se dividirá en distintas partes:

- Terminal autoservicio
- Aplicación mobile para clientes
- Web de toma de comandas en caja
- Terminal de recepción de comandas para cocina
- Backoffice web

2.6.1. Esquema general del sistema

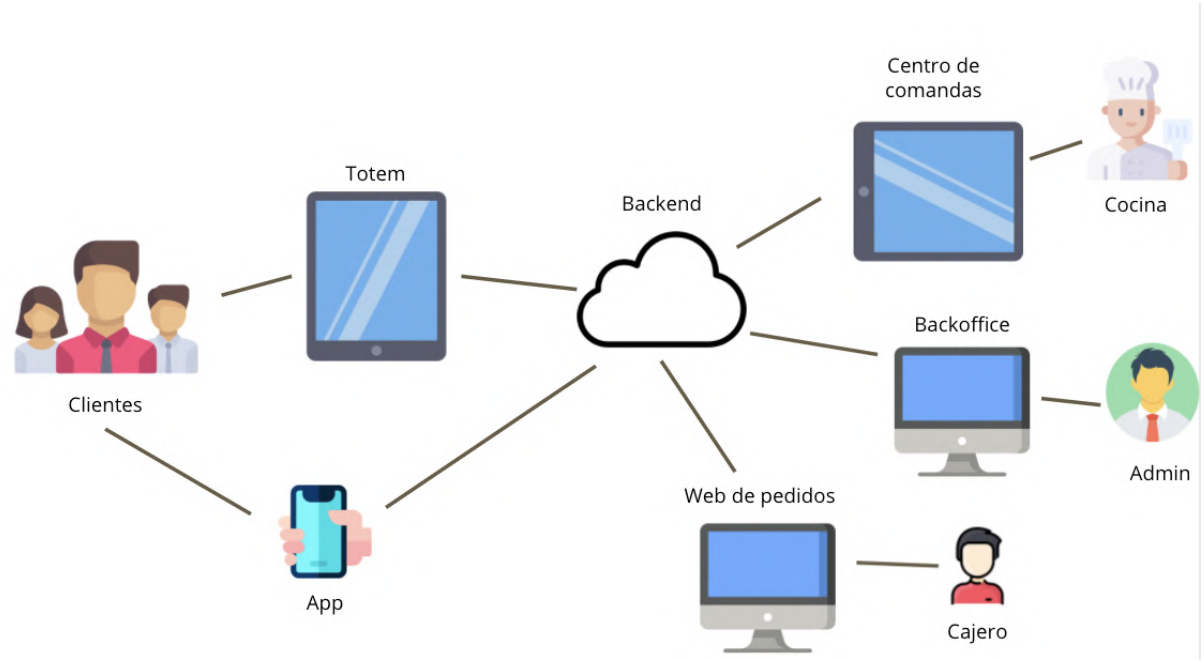


Figura 2: Esquema general del sistema

2.6.2. Terminal autoservicio

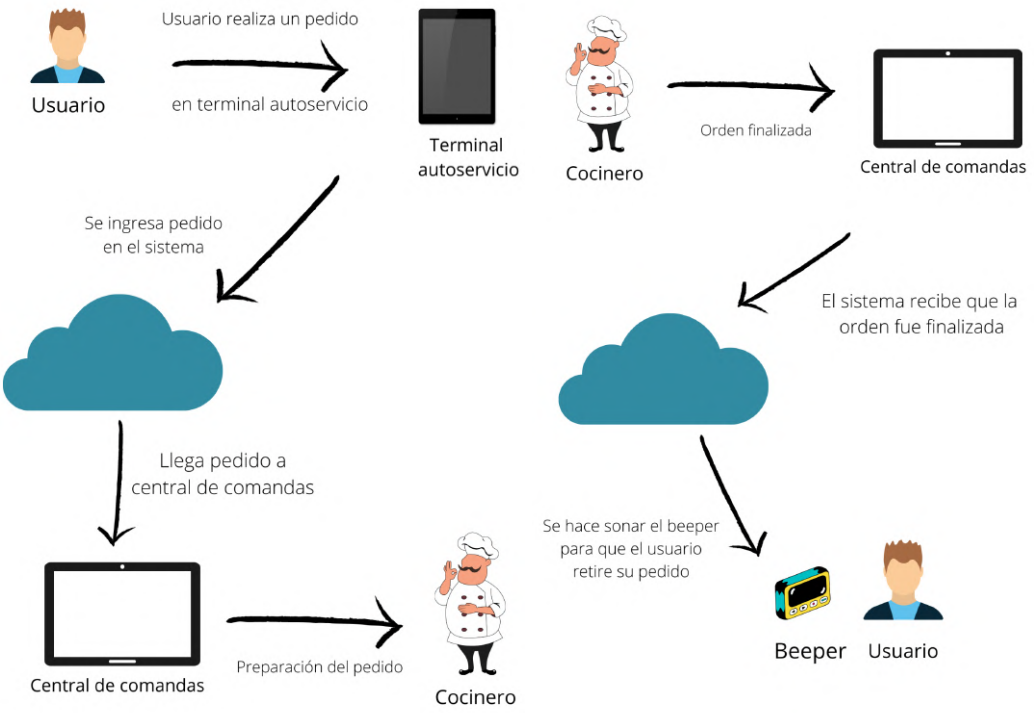


Figura 3 y 4: Flujo comanda terminal autoservicio

La terminal autoservicio o tótem es una aplicación desarrollada para tablets la cual el bar puede colocar en puntos estratégicos para que las personas que necesitan realizar un pedido puedan dirigirse y hacerlo desde ahí. Al costado de cada terminal de autoservicio se encuentran *beepers* disponibles para el usuario.

2.6.2.1. Selección de productos

Cuando un usuario se encuentra con la terminal autoservicio lo primero que puede ver es la carta completa de forma digital que se puede filtrar por categorías y seleccionar los productos que desea en las cantidades que desea. En cada producto puede ingresar notas si lo necesita (por ejemplo, para solicitar que se quite uno de los ingredientes del producto). A medida que se van seleccionando los productos, estos quedan de otro color de forma tal que el usuario sepa cuales ha agregado al carrito.

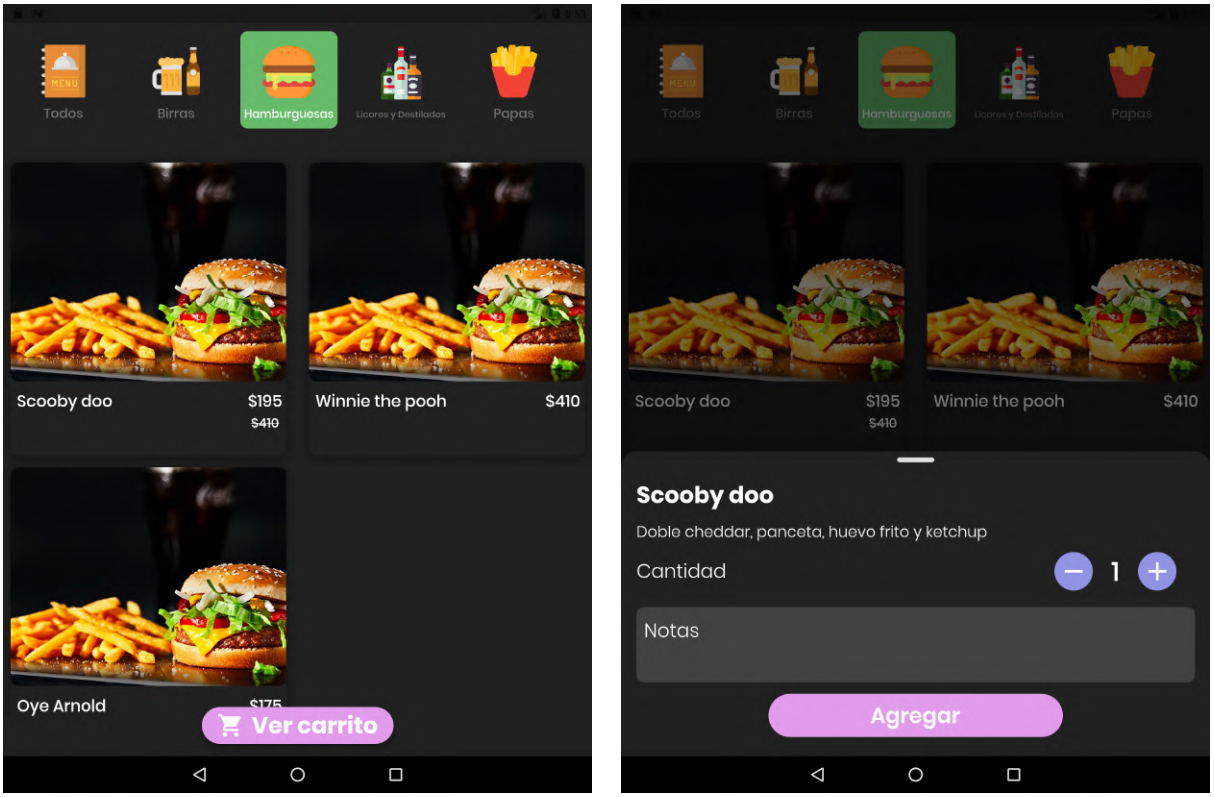


Figura 5 y 6: Selección de productos

2.6.2.2. Carrito

Cuando considera que ha terminado de agregar todos los productos que desea pedir, se dirige al carrito de compra en donde se le brinda al usuario nuevamente la

capacidad de poder modificar la cantidad de cada producto en específico o directamente quitar productos.

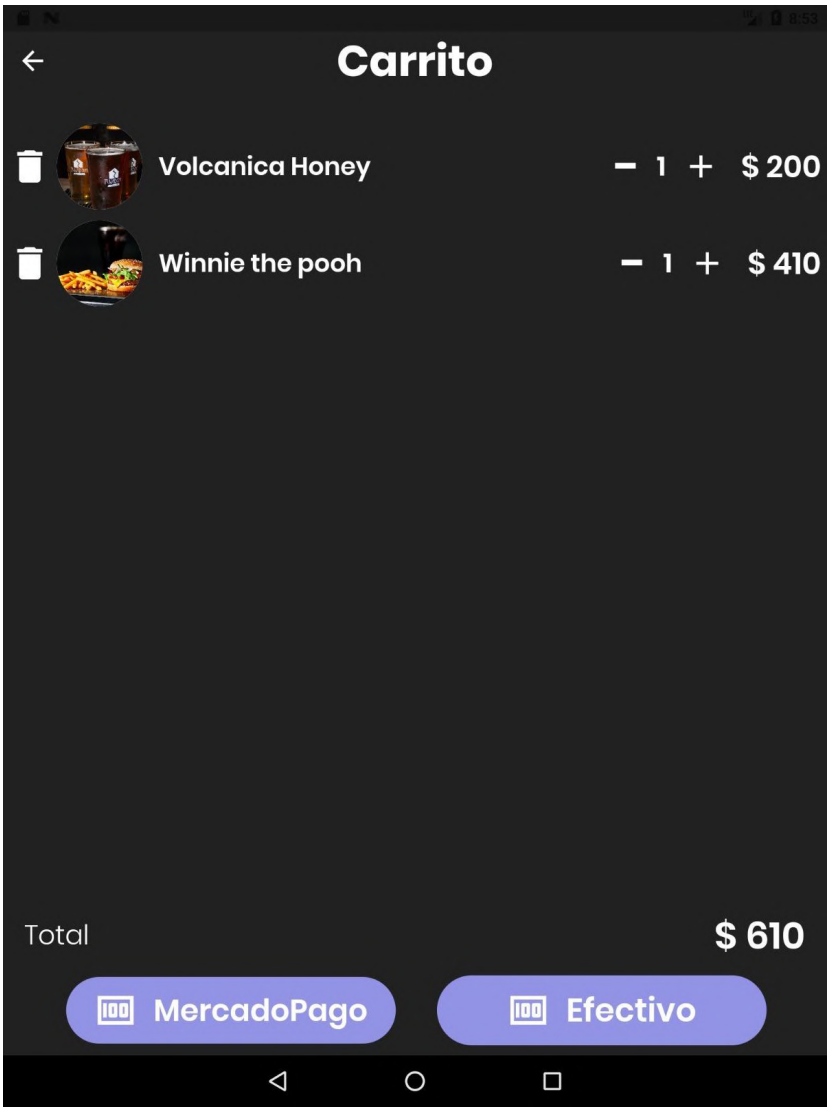


Figura 7: Carrito

2.6.2.3. Pago

Una vez que el usuario decide que ha agregado todos los elementos necesarios al carrito, prosigue a pagar su compra, para lo que se le ofrecen dos métodos, pago por Mercado Pago o efectivo.

- Pago a través de Mercado Pago

Cuando el usuario selecciona la opción de pagar vía Mercado Pago aparece una pantalla en donde se le pide que ingrese el número de *beeper* (el cual deberá agarrar del costado de la terminal autoservicio, y que servirá para relacionar un pedido a un cliente), luego presiona el botón "Terminar" y en la pantalla aparecerá un código QR que deberá escanear.

El código QR es generado por Mercado Pago a través de una *api* el cual es de único uso y referencia específicamente a ese pedido. De esta manera cuando el usuario escanea el código puede acceder a la información correspondiente y el precio queda ingresado automáticamente haciendo que el proceso de pago sea de un solo paso.

Una vez que el código fue escaneado la comanda es enviada a cocina para ser preparada.

Figura 8: Pago por Mercado Pago

- Pago en efectivo

Cuando el cliente selecciona la opción de pagar en efectivo, al igual que en la opción anterior, aparece la pantalla de ingresar el *beeper,* una vez que el usuario presiona el botón “Terminar” la comanda es enviada a cocina para ser preparada y el usuario debe dirigirse en ese momento a la caja para abonar.

En ambos flujos el usuario será notificado a través del *beeper* que su pedido se encuentra listo.

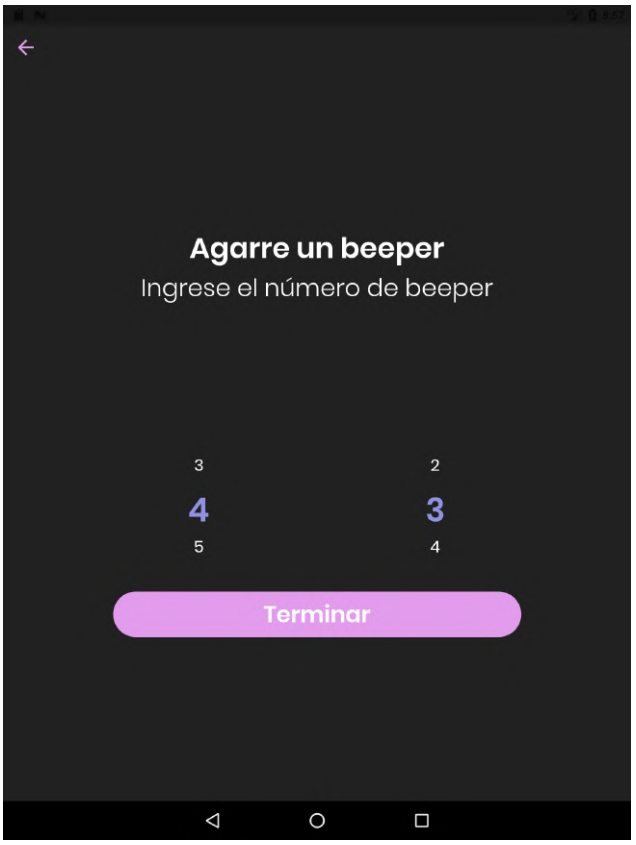


Figura 9: Selección de beeper

2.6.3 Aplicación mobile para clientes

La aplicación *mobile* fue desarrollada para contemplar a los usuarios que no desean utilizar la terminal autoservicio y desean hacer el pedido desde sus propios celulares.

2.6.3.1. Login

La primera pantalla que se encuentra al abrir la aplicación es el *login* en donde los usuarios tienen la opción de registrarse, recuperar la contraseña o iniciar sesión.

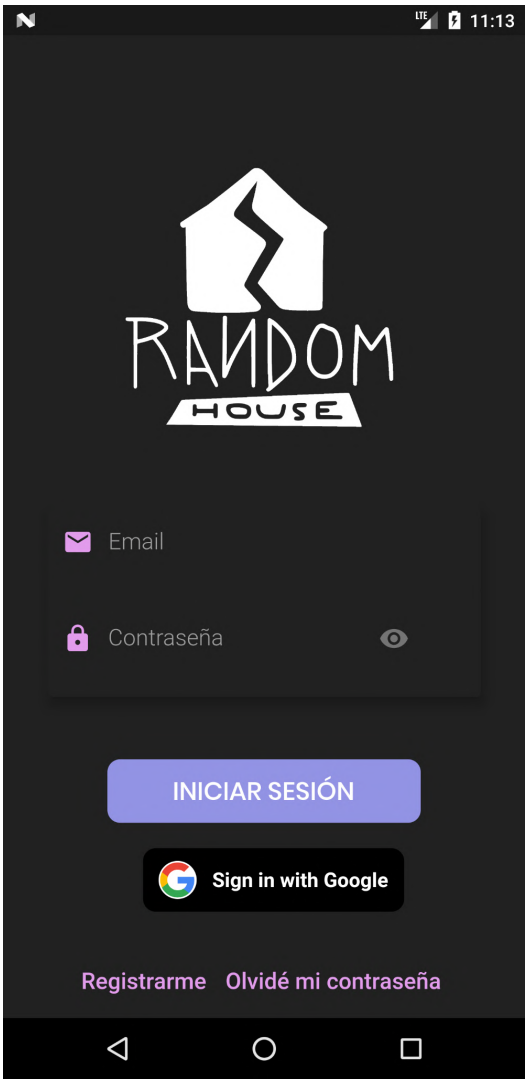


Figura 10: Login

2.6.3.2. Recuperar contraseña

En el caso que el usuario desee recuperar la contraseña se le mostrará la siguiente pantalla.

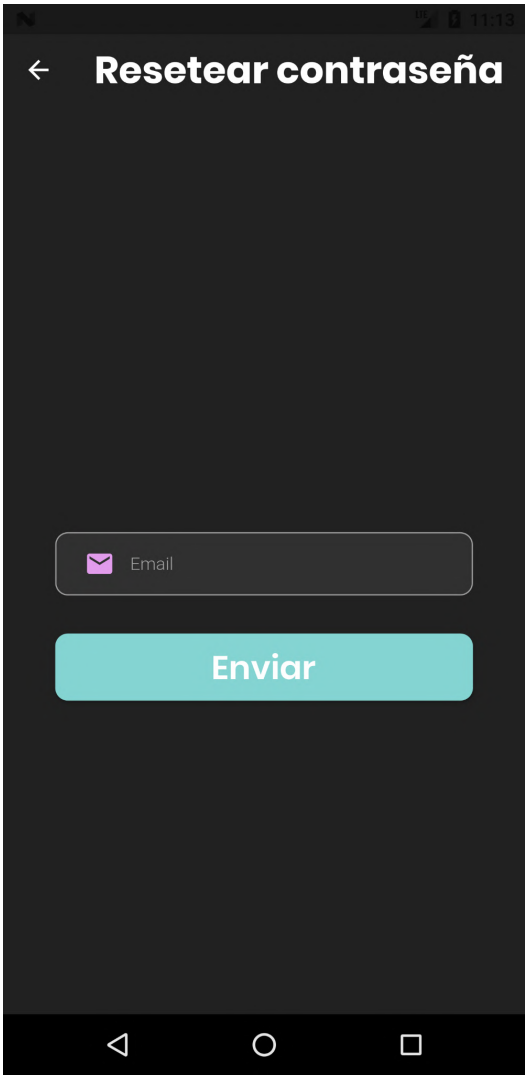


Figura 11: Recuperar contraseña

Una vez que el usuario ingresa su email, se le envía una contraseña de único uso para que el usuario pueda iniciar sesión y luego cambiarla por una nueva.

Figura 12: Email recuperar contraseña

2.6.3.3. Pedidos y carrito

Dentro de esta aplicación los usuarios pueden acceder a la carta de la misma manera que se puede acceder desde la terminal autoservicio, una vez que se accede al carrito se encuentra la opción de si se tiene un cupón.

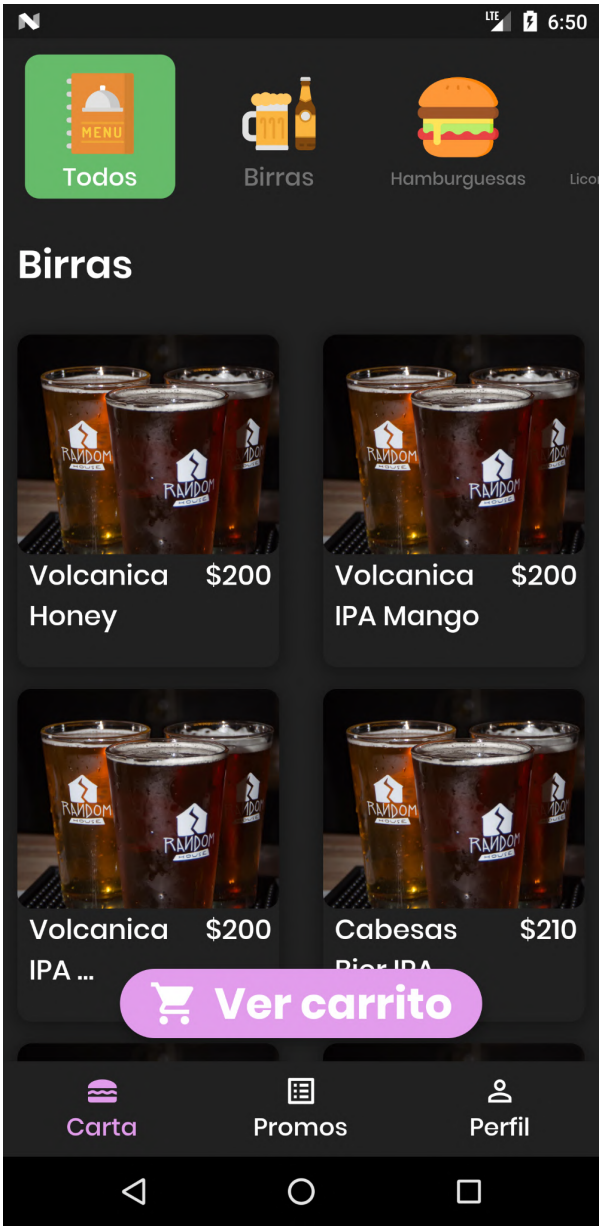


Figura 13: Menú app móvil

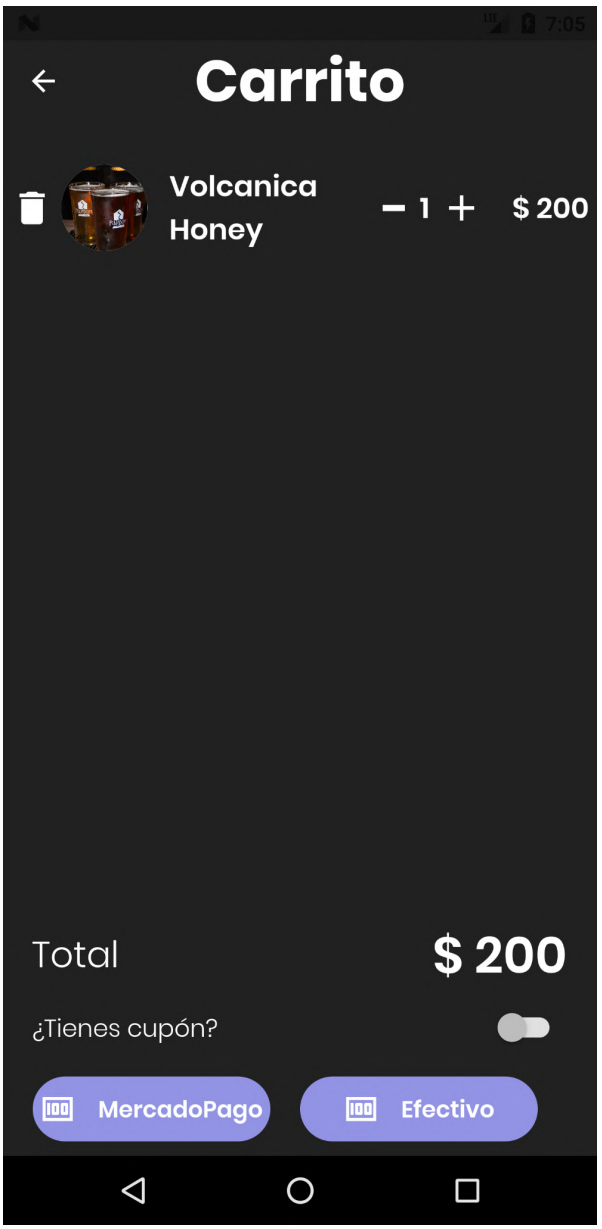


Figura 14: Carrito app móvil

2.6.3.4. Cupón

En caso de seleccionar la opción de tener cupón, al momento de elegir pagar con Mercado Pago aparece la pantalla para ingresar el cupón. Cuando se ingresa un cupón y se presiona el botón “Aplicar” se muestra el nuevo precio con descuento.

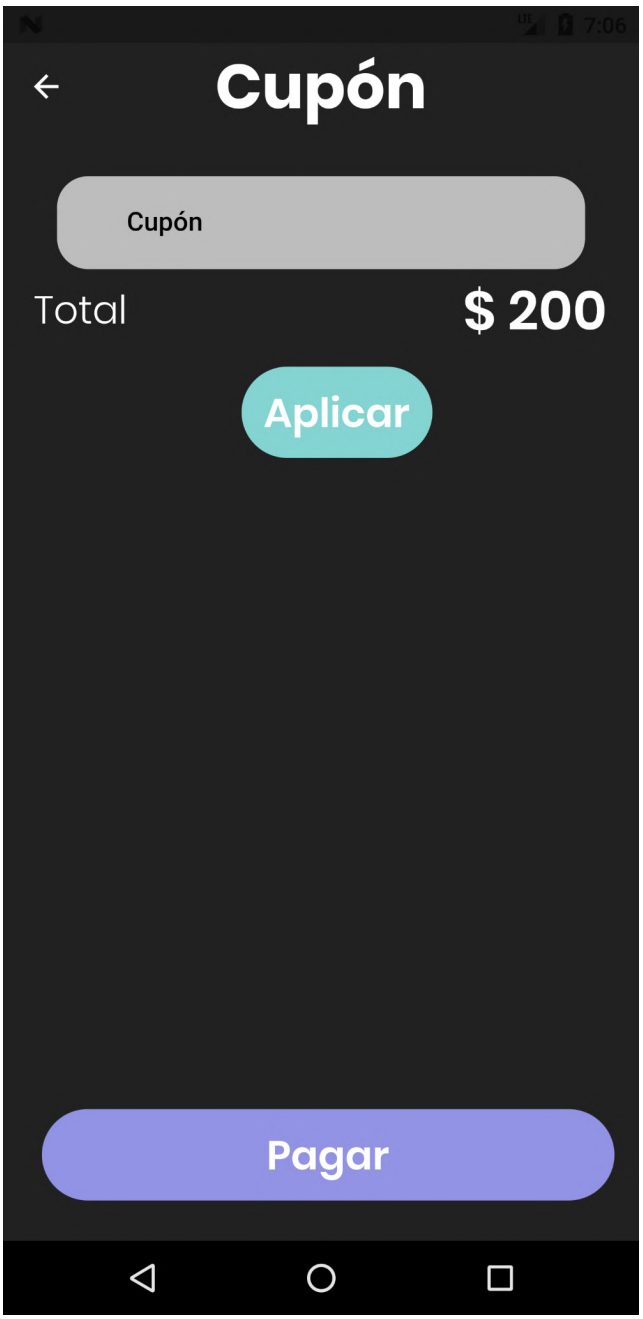
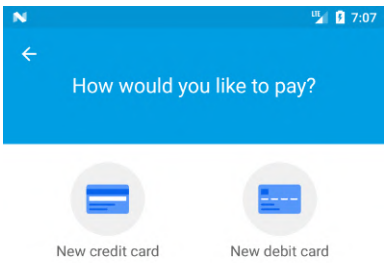


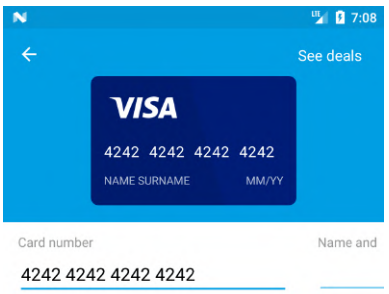
Figura 15: Ingresar cupón app móvil

2.6.3.5. Pago

Cuando el usuario presiona el botón “Pagar” se abre el *checkout* de Mercado Pago en donde el usuario puede visualizar la cantidad que debe pagar y tiene la opción para pagar con tarjeta de crédito o débito.

En el caso de no tener un cupón se cobrará la totalidad del precio ya sea vía Mercado Pago o vía efectivo.





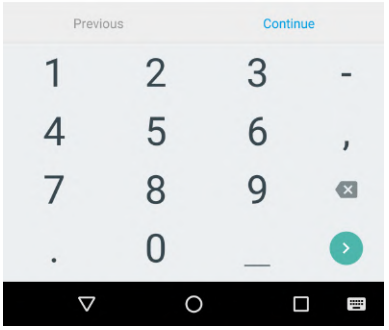
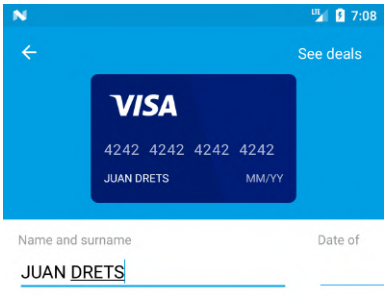
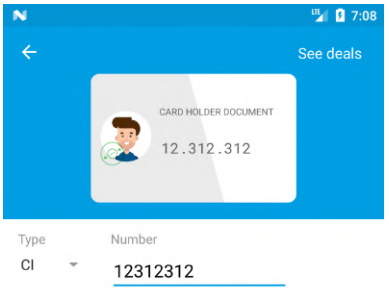
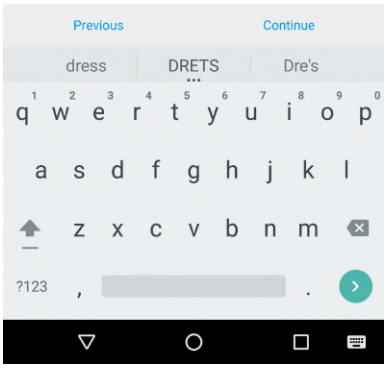


Figura 16 y 17: Asociar tarjeta a mercado pago app móvil (parte 1)







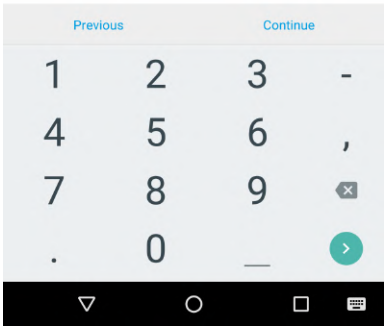


Figura 18 y 19: Asociar tarjeta a mercado pago app móvil (parte 2)

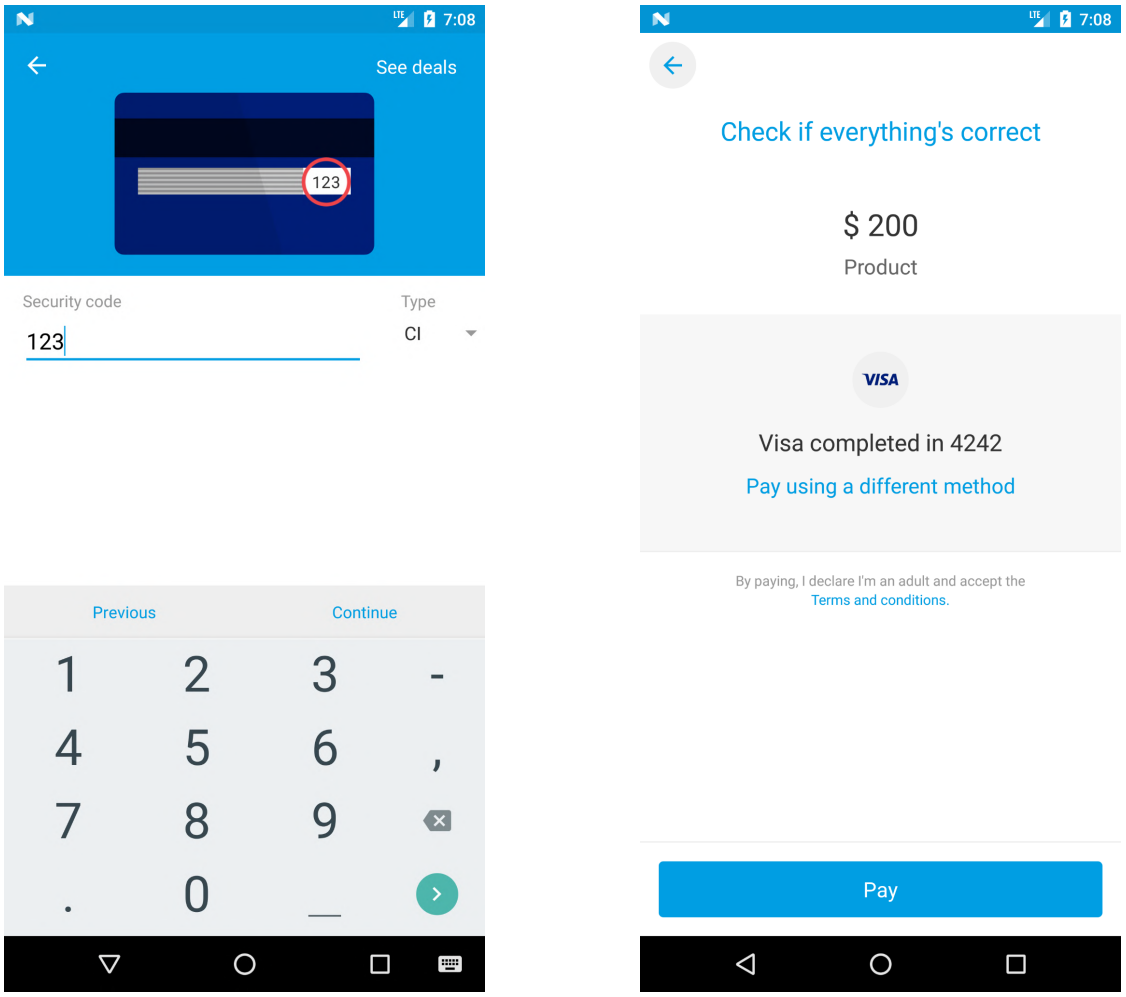


Figura 20 y 21: Asociar tarjeta a mercado pago app móvil (parte 3)

2.6.3.6. Promociones

La segunda pestaña es donde se encuentran las promociones. Dicha sección se separa en dos, promociones comunes y promociones destacadas, la diferencia entre las mismas está en la disposición en la pantalla. Las promociones se muestran en forma de imágenes en donde cada una de ellas indica los detalles de la misma.

Estas promociones son aplicadas por el sistema de forma automática, por ejemplo, si dentro de las promociones se encuentra una imagen en donde se indica que hay 50% de descuento en la pizza 3 quesos entonces en los precios del menú se verá el precio original y el precio con descuento. Si el usuario selecciona ese producto, cuando se dirija a la pantalla del carrito podrá ver la suma de los precios, que en caso de que haya productos en promoción se sumará ese precio al total en vez del

precio original de los mismos. Esto se hace como forma de fomentar a los usuarios a usar el sistema.

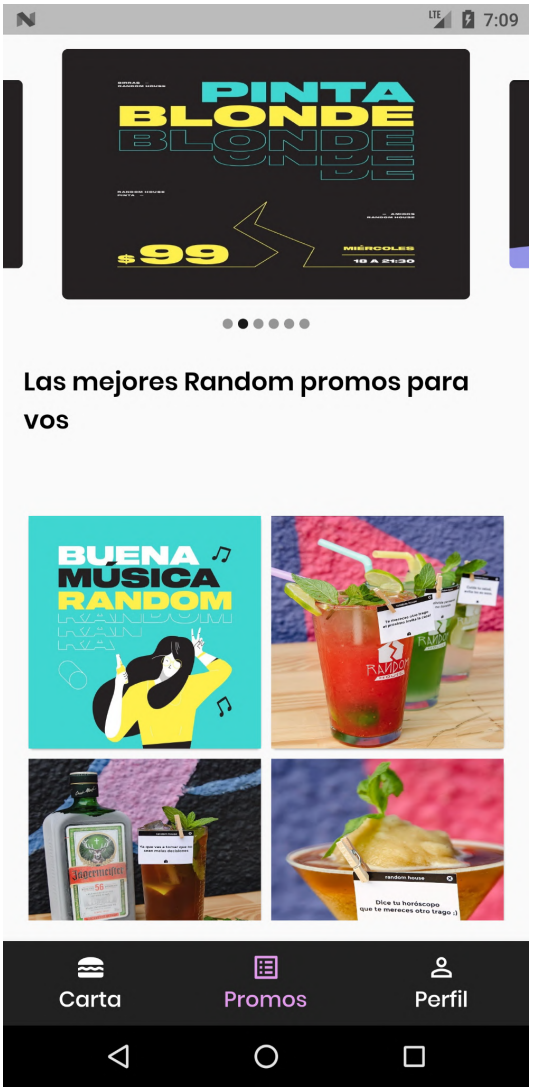


Figura 22: Promociones app móvil

2.6.3.7. Perfil

La tercera pestaña es donde se encuentra toda la información referente al usuario. Dentro de esta se encuentran los datos del perfil del usuario, así como también la opción para editar dichos datos, que se accede presionando el botón de “Editar perfil”.

Luego, debajo de los datos del usuario se encuentran los pedidos sin pagar. De esta manera cuando el usuario desee pagar sus pedidos del día, la persona encargada de cobrarlos podrá corroborar los pedidos. Dentro del listado el usuario puede

acceder a cada uno de sus pedidos por separado, es decir si presiona un pedido en específico podrá ver el desglose de productos pedidos con sus precios correspondientes.

El usuario tiene un botón en el cual puede acceder al historial de pedidos. Para cada uno de estos se muestra el id del mismo por si ocurre algún problema con cierto pedido el usuario pueda realizar el reclamo y los empleados puedan chequear en el sistema. Al igual que con los pedidos sin pagar, el usuario podrá acceder a cualquier pedido en específico y ver los productos correspondientes con sus precios y el total de la orden. Para cada pedido se encuentra un botón en donde el usuario podrá repetirlo, por lo tanto, con un solo click podrá acceder al carrito con los productos precargados del pedido anterior que seleccionó.

Por último en esta pantalla se encuentra el botón "*Logout*" en donde se podrá cerrar la sesión.

Figura 23: Perfil de usuario app móvil

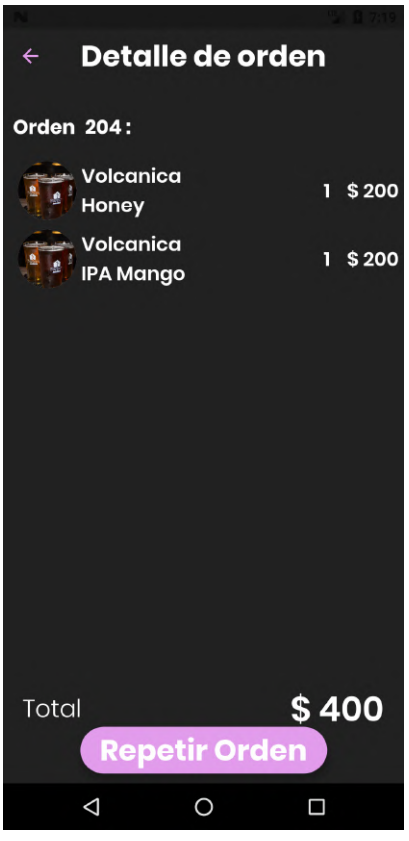


Figura 24: Detalle de orden app móvil

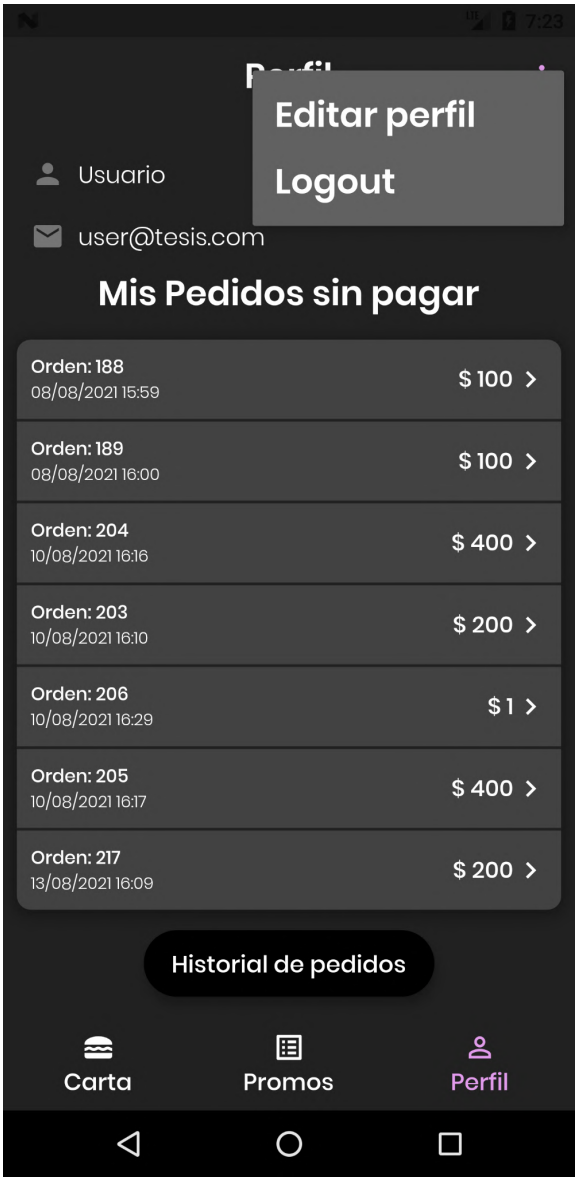


Figura 25: Opciones de perfil app móvil

Cuando un pedido está listo, es decir en la terminal de comandas en cocina lo mueven a la columna “Completados” al usuario que realizó el pedido le llega una *push notification* indicando que su pedido está listo.

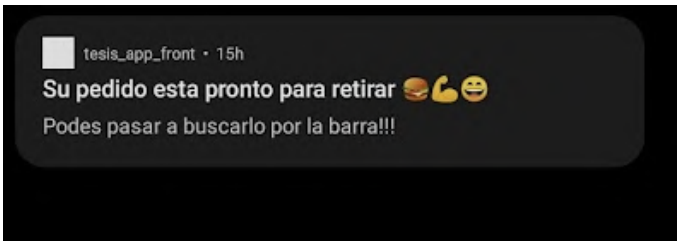
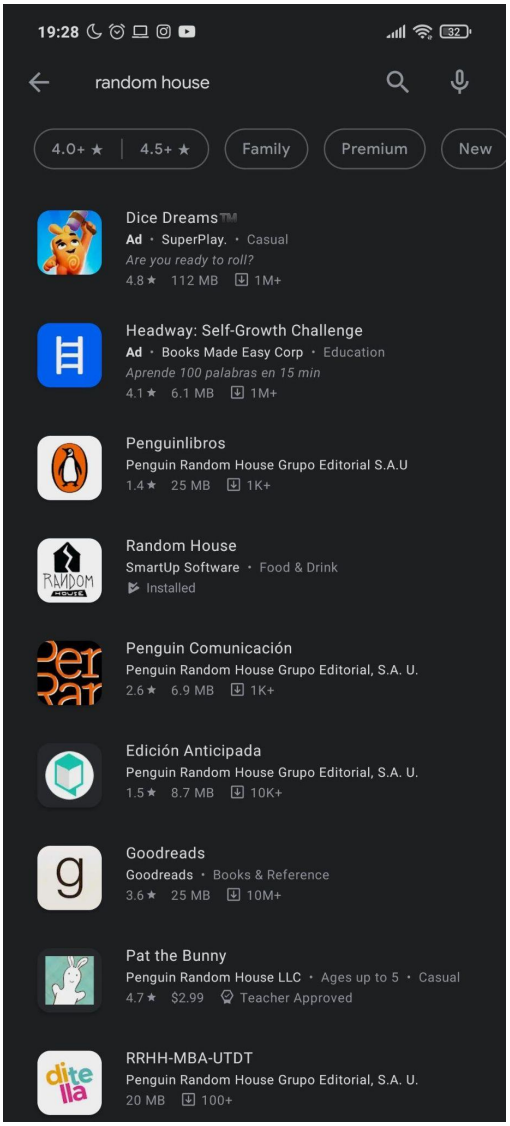


Figura 26: Notificación retirar pedido app móvil

Dicha app se encuentra disponible en Play Store [46] para su descarga.



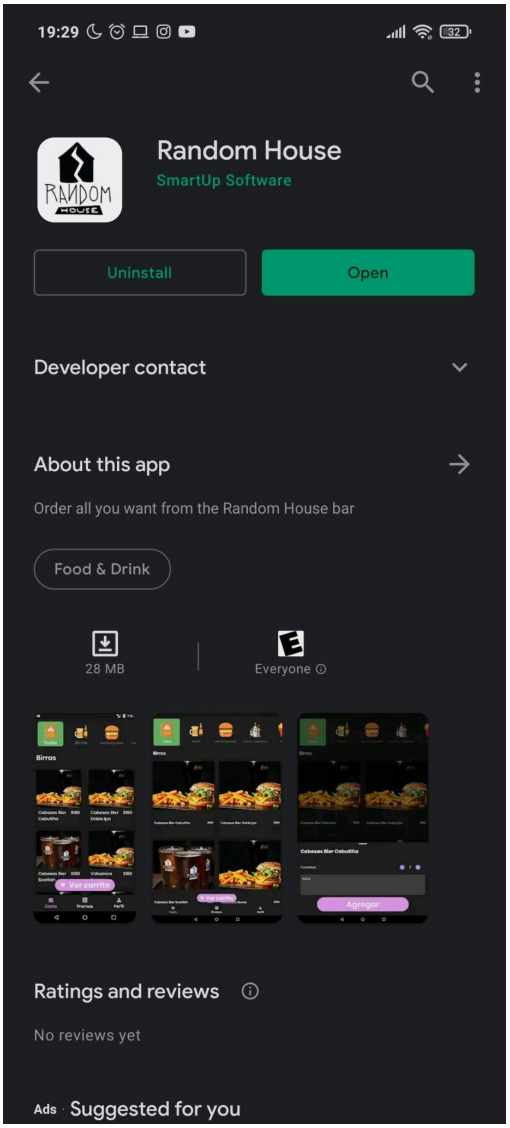


Figura 27 y 28: Presencia en Play Store

2.6.4. Web de toma de comandas en caja

Como se mencionó anteriormente, los usuarios tienen la opción de realizar los pedidos en persona desde la caja. Es por esto que se desarrolló una web para cubrir también este escenario.

La solución consiste en una web que será accedida por la persona encargada de tomar los pedidos desde la caja, de esta manera, cuando un usuario se dirige a la caja, el mismo es tomado e ingresado en el sistema y la comanda es enviada a cocina de forma automática.

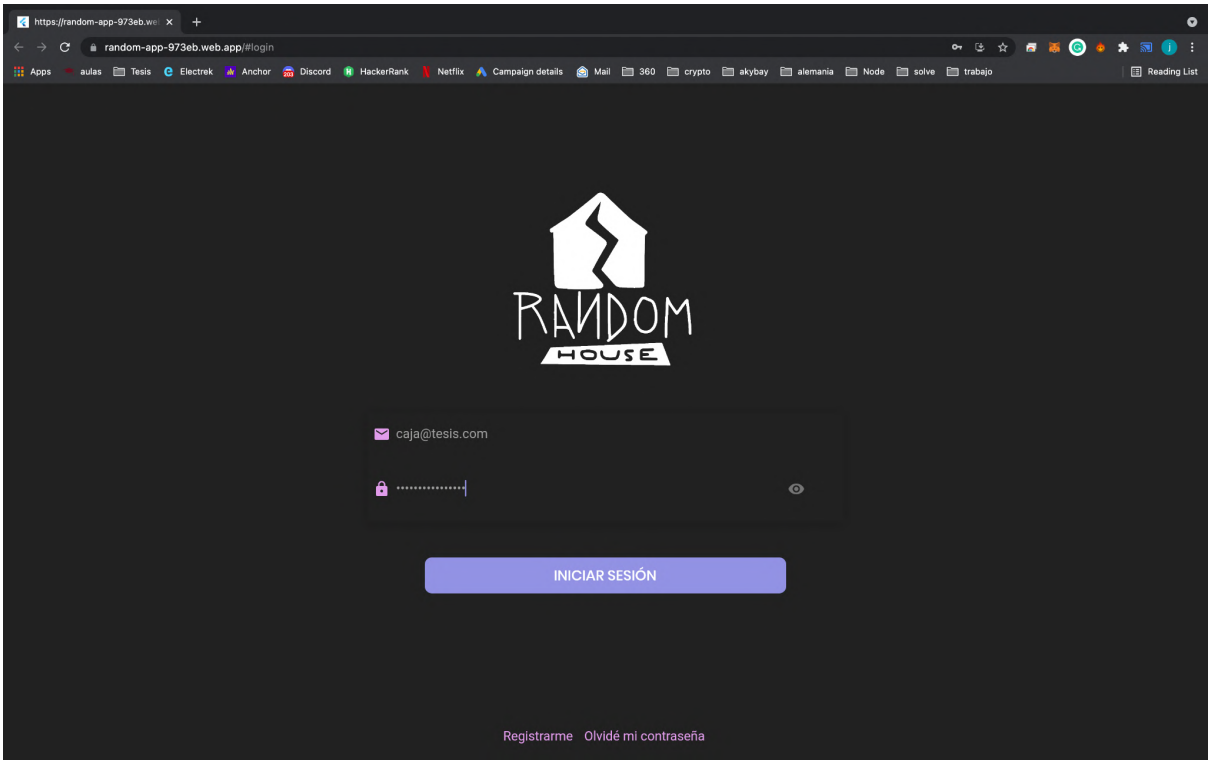


Figura 29: Login web

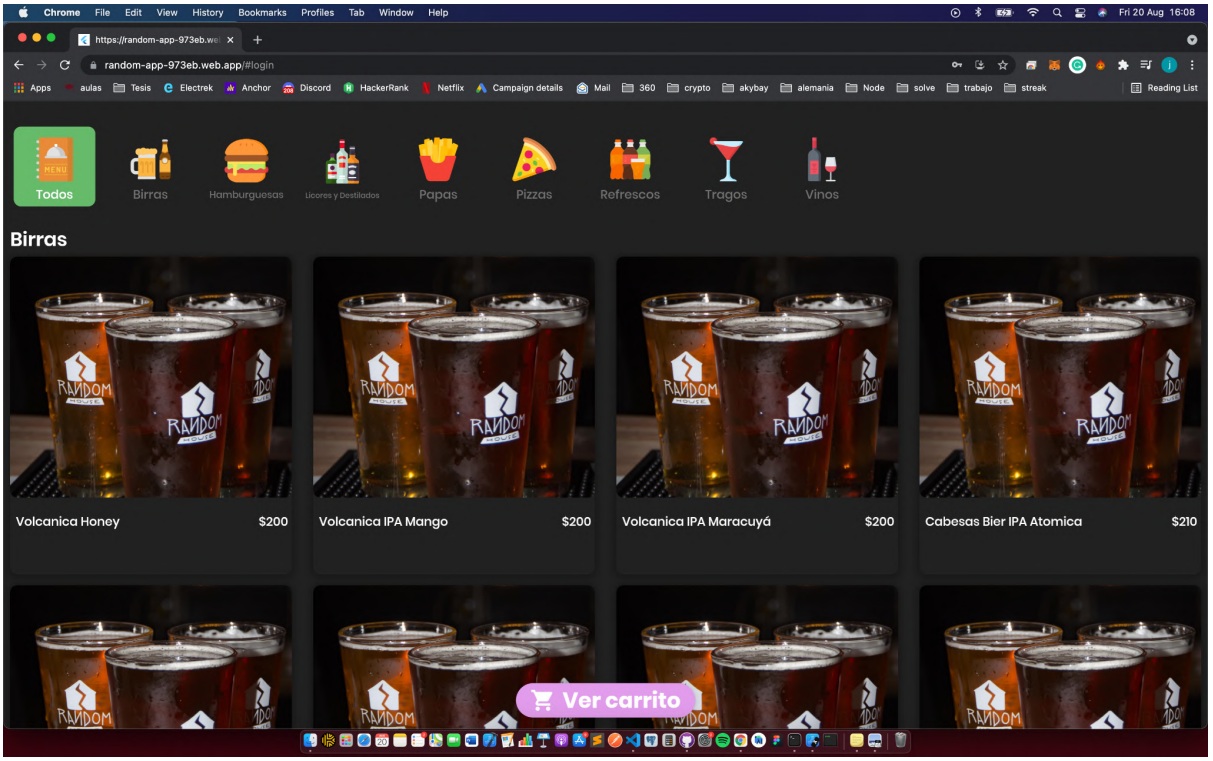


Figura 30: Home web

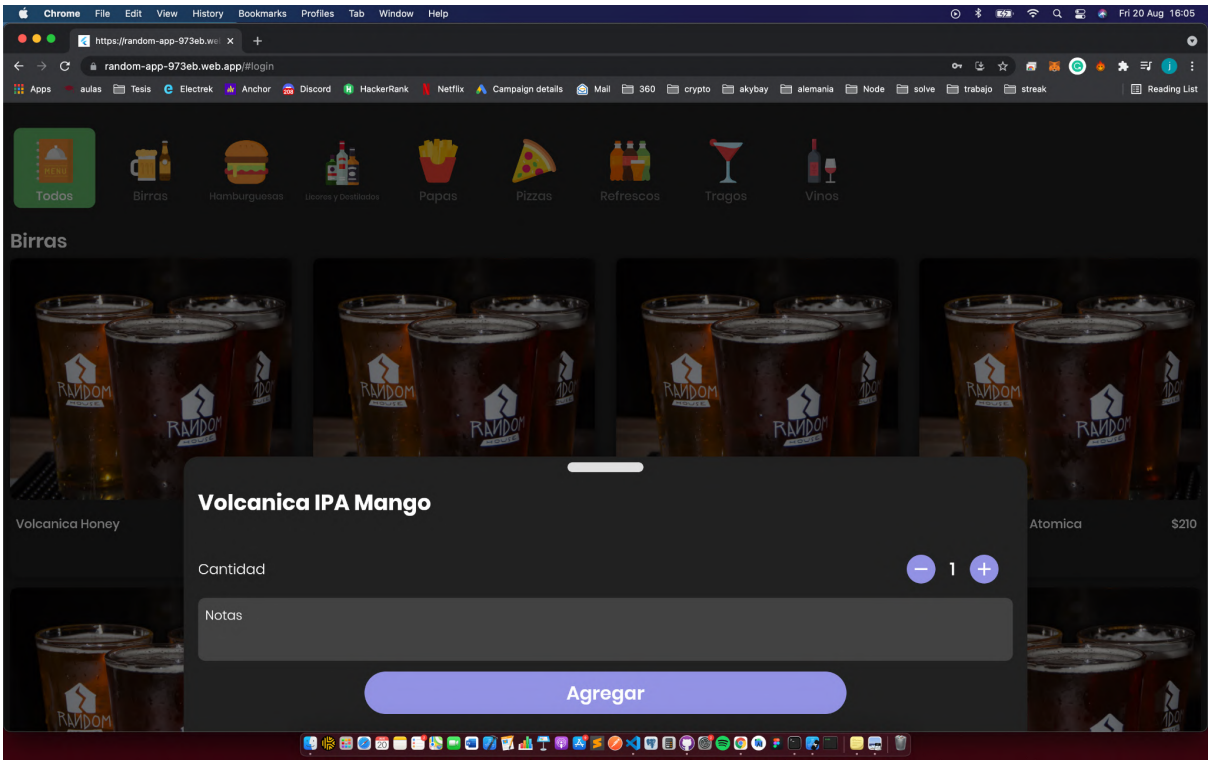


Figura 31: Agregar producto web

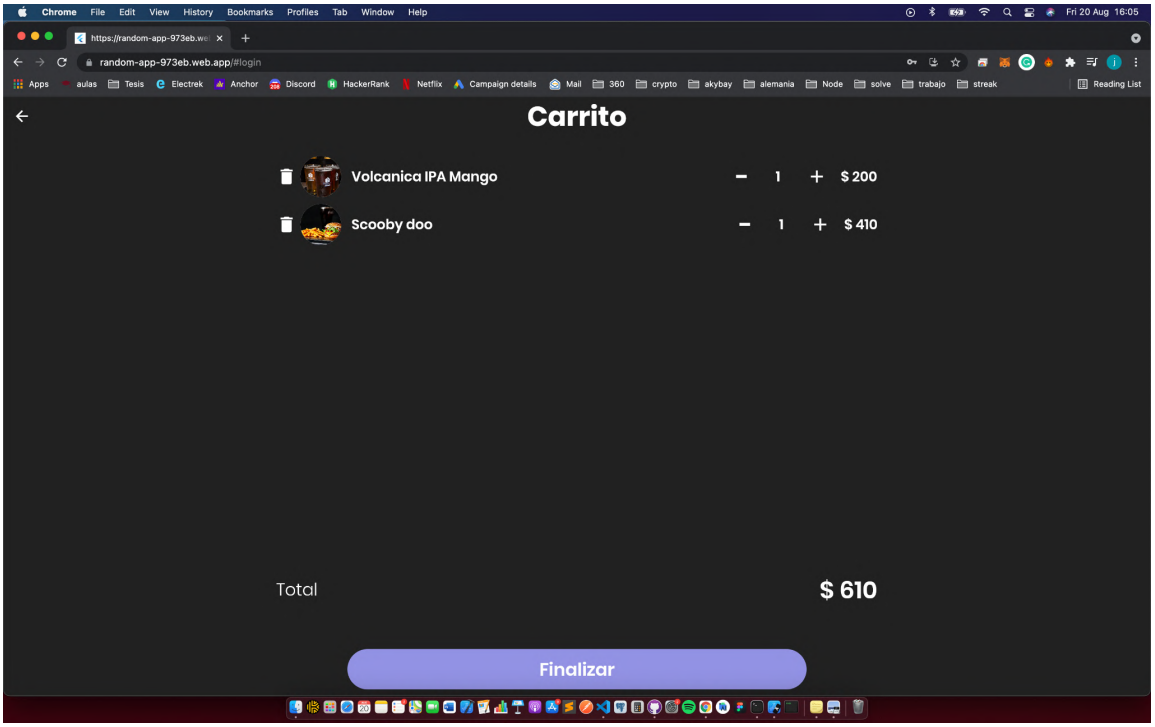


Figura 32: Carrito web

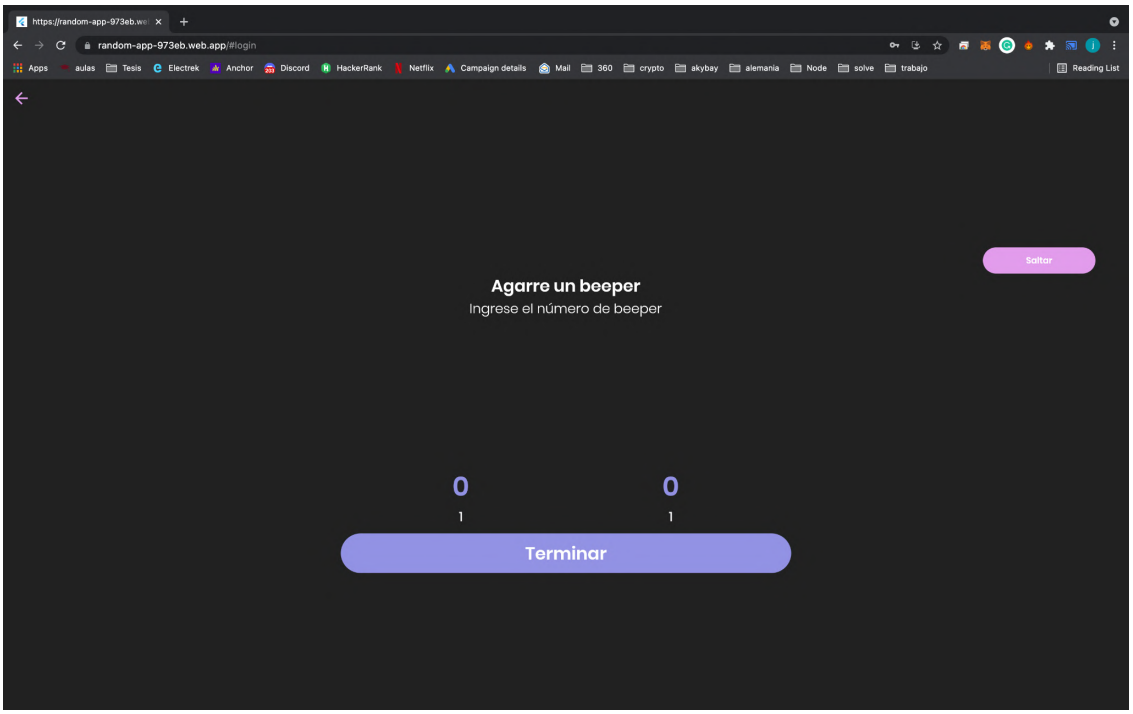


Figura 33: Selección beeper web

2.6.5. Terminal de recepción de comandas para cocina

La solución cuenta con otra parte fundamental, un sistema de recepción de comandas en cocina.

Dentro de la pantalla principal del sistema de comandas nos encontramos con dos columnas:

La primera es la columna de pedidos pendientes en donde se puede visualizar de manera sencilla cada una de las comandas con los siguientes datos: productos pedidos y cantidades correspondientes, *beeper* asociado a la comanda, hora de realizada la comanda y tiempo de espera que lleva el cliente desde que realizó el pedido.

La segunda es la columna de los pedidos en preparación, es decir, cuando un cocinero recibe un pedido nuevo y comienza a cocinarlo este debe (por medio de *drag and drop*) mover ese pedido hacia la columna “en preparación”. De esta manera podrá tener un listado de todos los productos que la cocina se encuentra preparando en ese momento.

Luego hay otra pantalla que se accede presionando el botón “Ver finalizados” en donde nuevamente hay dos columnas.

En la primera columna se encuentran los pedidos completados, es decir todos los pedidos que llegaron, luego fueron preparados para posteriormente ser marcados como completados. De esta manera se tiene un registro de todos los trabajos realizados en cocina en caso de haber algún tipo de problema o consulta.

En la segunda columna se encuentran los pedidos que por algún motivo han sido cancelados. Un ejemplo de este flujo sería si por algún motivo la persona encargada avisa a cocina que cierto pedido no debe ser ejecutado entonces en cocina lo cancelan e igualmente queda el registro de absolutamente todos los pedidos.

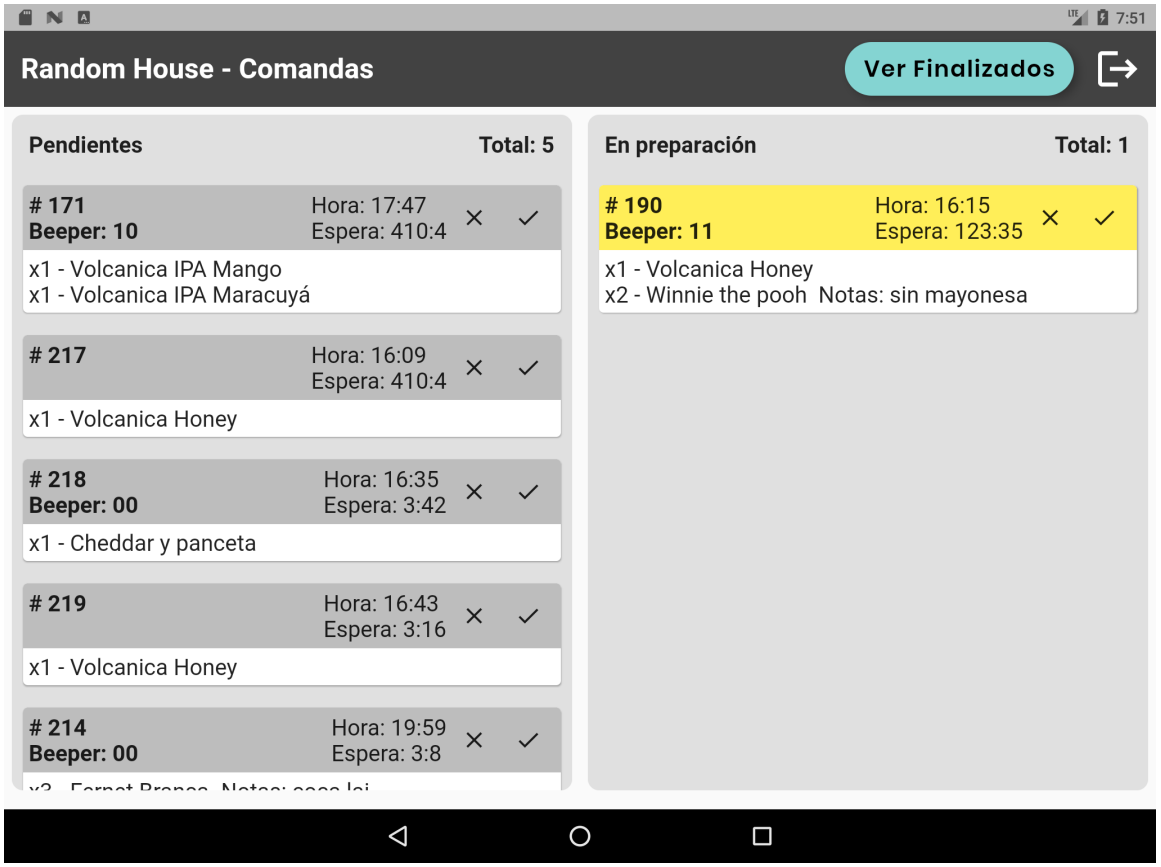


Figura 34: Pantalla de comandas (parte 1)

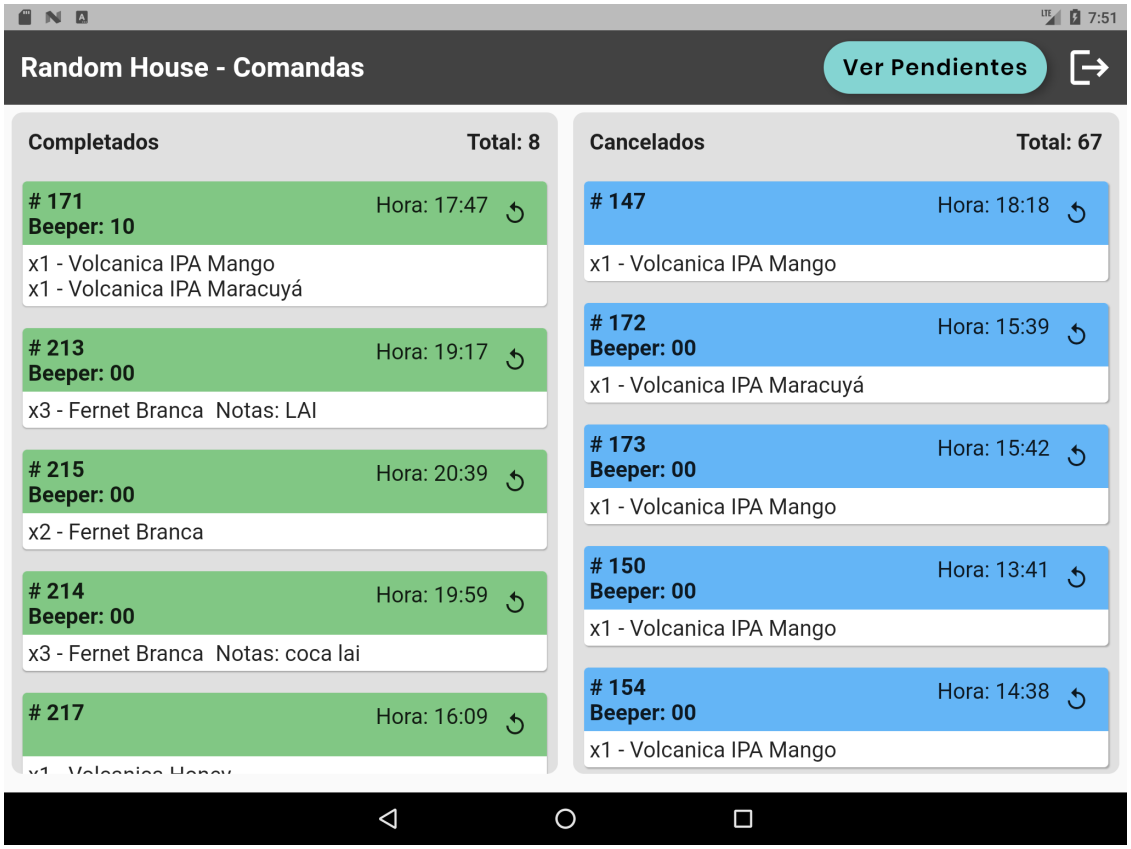


Figura 35: Pantalla de comandas (parte 2)

2.6.6. Backoffice web

Como último componente del sistema se desarrolló un *backoffice* web el cual es utilizado por las personas encargadas del bar en donde se puede gestionar todo sobre la solución.

El back office cuenta con las siguientes partes:

1. Menú
2. Dashboard
3. Usuarios
4. Categorías
5. Categorías de stock
6. Productos
7. Comandas
8. Comandas efectivo
9. Promociones
10. Clientes
11. Cupones

2.6.6.1. Menú

Se integró dentro del *backoffice* el menú web para que en la caja puedan realizar los pedidos al tener en una sola web una herramienta integral en donde se pueda gestionar todo.

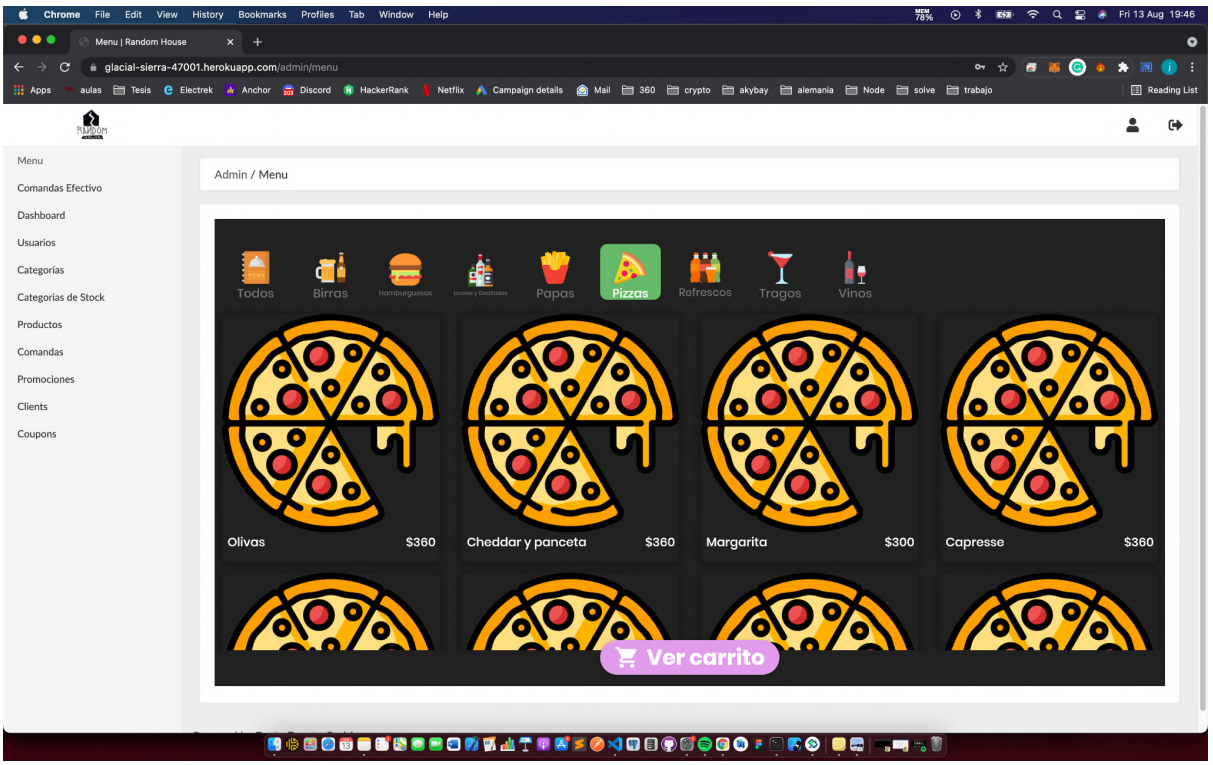


Figura 36: Menú Backoffice

2.6.6.2. Dashboard

En esta pantalla se encuentran gráficas e información la cual el cliente indicó que era de valor. Esta información está conformada por: Productos más vendidos, ventas por día, ventas por plataforma, promociones más utilizadas y ventas por plataforma.

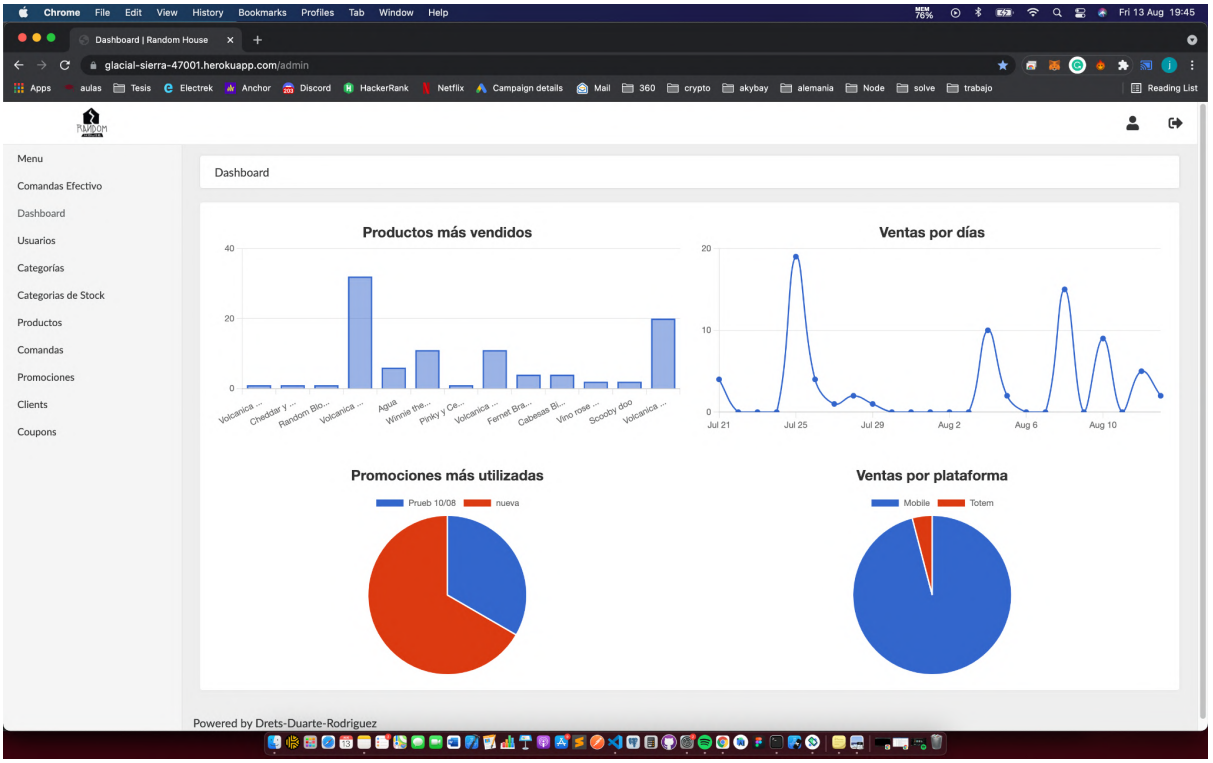


Figura 37: Dashboard Backoffice

2.6.6.3. Usuarios

Se muestra el listado de todos los usuarios registrados que tiene el sistema. En caso de ser necesario, el administrador cuenta con los permisos para poder agregar uno nuevo o eliminar alguno ya existente.

2.6.6.4. Categorías

Se muestran todas las categorías que se encuentran en la carta. De una forma sencilla el usuario puede ver, modificar o eliminar una categoría.

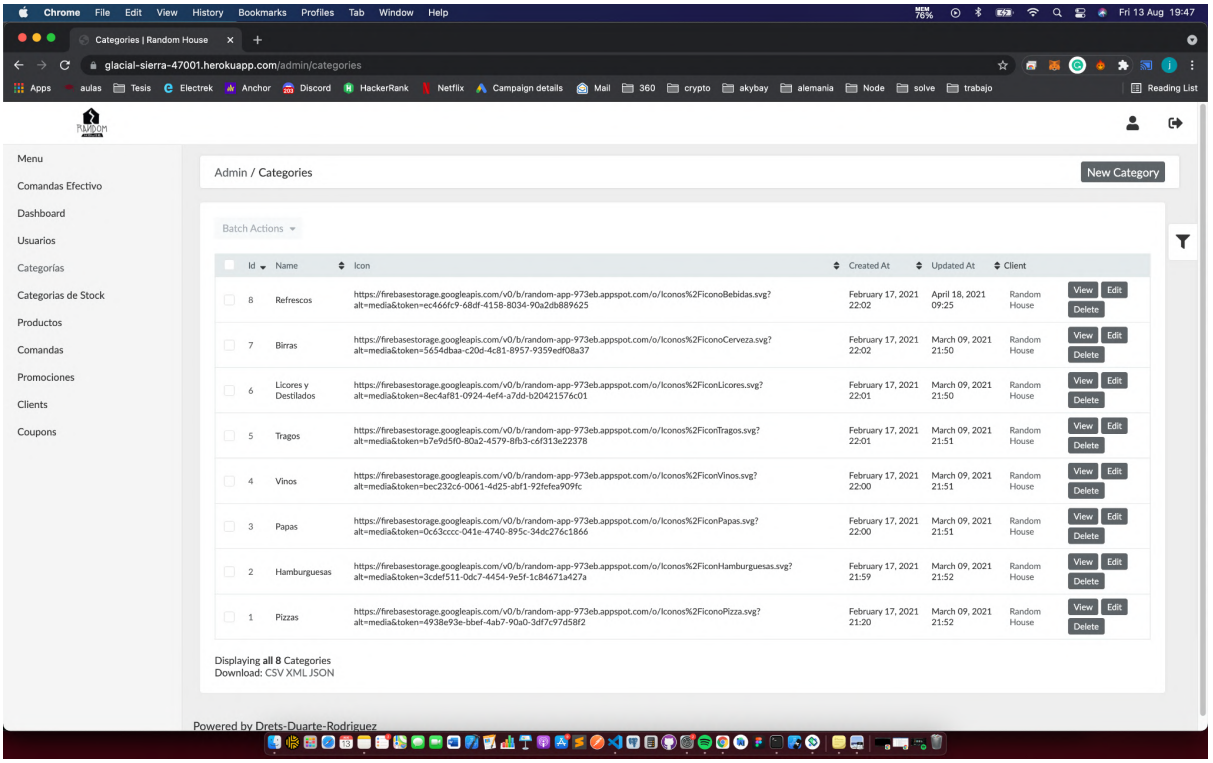


Figura 38: Categorías Backoffice

2.6.6.5. Categorías de stock

En esta pestaña se pueden ver, modificar o eliminar las categorías de *stock*. Las categorías de *stock* son, como dice la palabra, categorías en donde al crear una de estas, se ingresa un nombre, una cantidad de *stock* y los productos que lo descuentan. Por ejemplo: cuando el bar compra panes de hamburguesas entonces debería crear (la primera vez) la categoría de *stock* “panes de hamburguesas”, luego agregar la cantidad de *stock* (si compro 100 panes debería ingresar 100) y por último elegir los productos que en su composición se encuentran los “panes de hamburguesa” entonces en este caso debería elegir todas las hamburguesas. De esta manera, por cada pedido que se haga de hamburguesas se descontará una unidad del *stock* “panes de hamburguesa”.

De esta manera las dueñas del bar pueden tener un control del *stock* y saber cuándo alguno de los productos se encuentra bajo de *stock* y reponerlo.

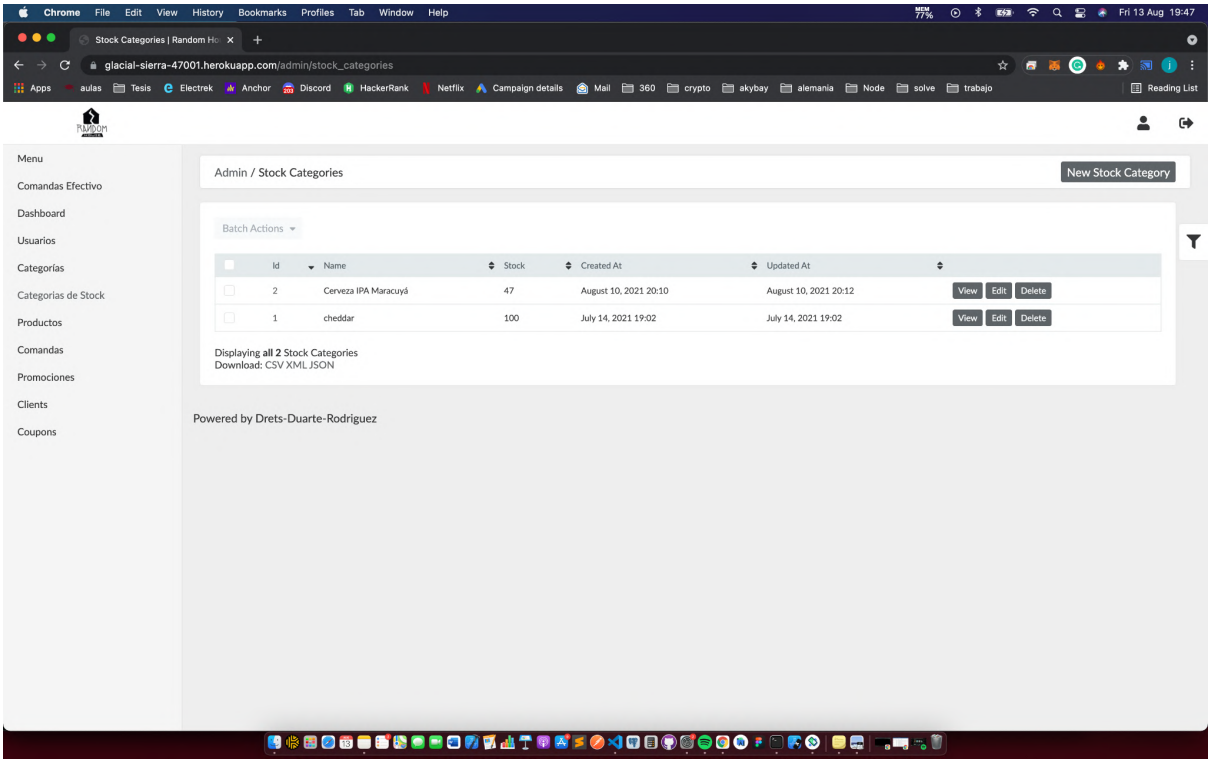


Figura 39: Categorías de stock Backoffice

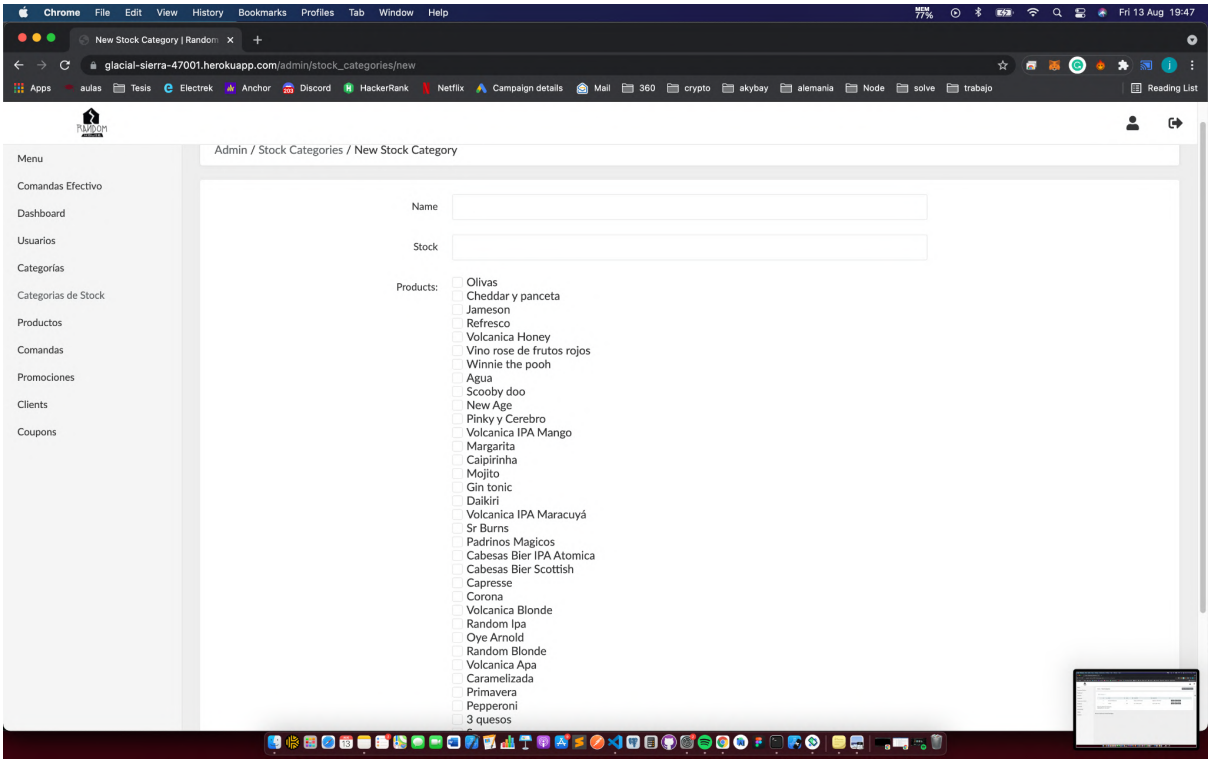


Figura 40: Agregar categorías de stock Backoffice

2.6.6.6. Productos

Aquí se encuentra un listado de todos los productos de la carta y de una forma sencilla el usuario puede ver, modificar o eliminar un producto.

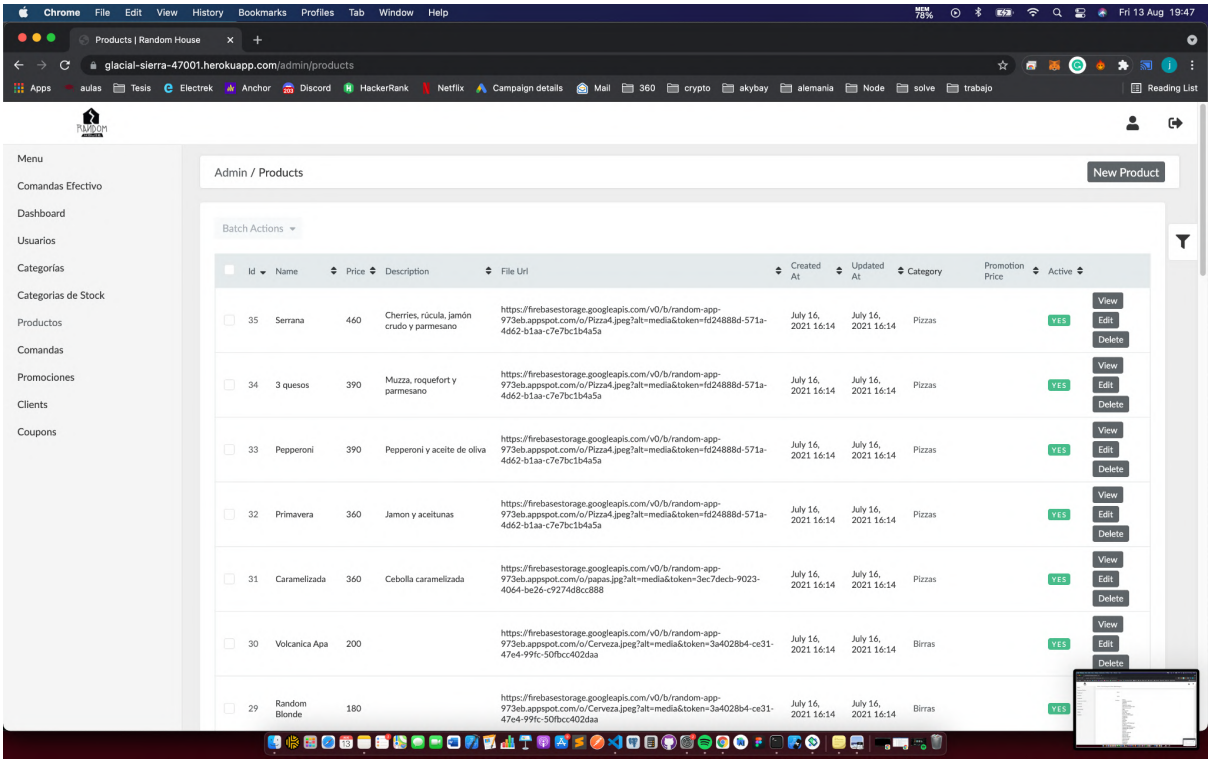


Figura 41: Productos Backoffice

2.6.6.7. Comandas

Aquí se encuentra el listado de todas las comandas recibidas por el sistema que fueron pagadas por Mercado Pago. Para cada una de estas se muestra el estado del pago, si ha sido pedido desde la terminal de autoservicio y el estado de la comanda.

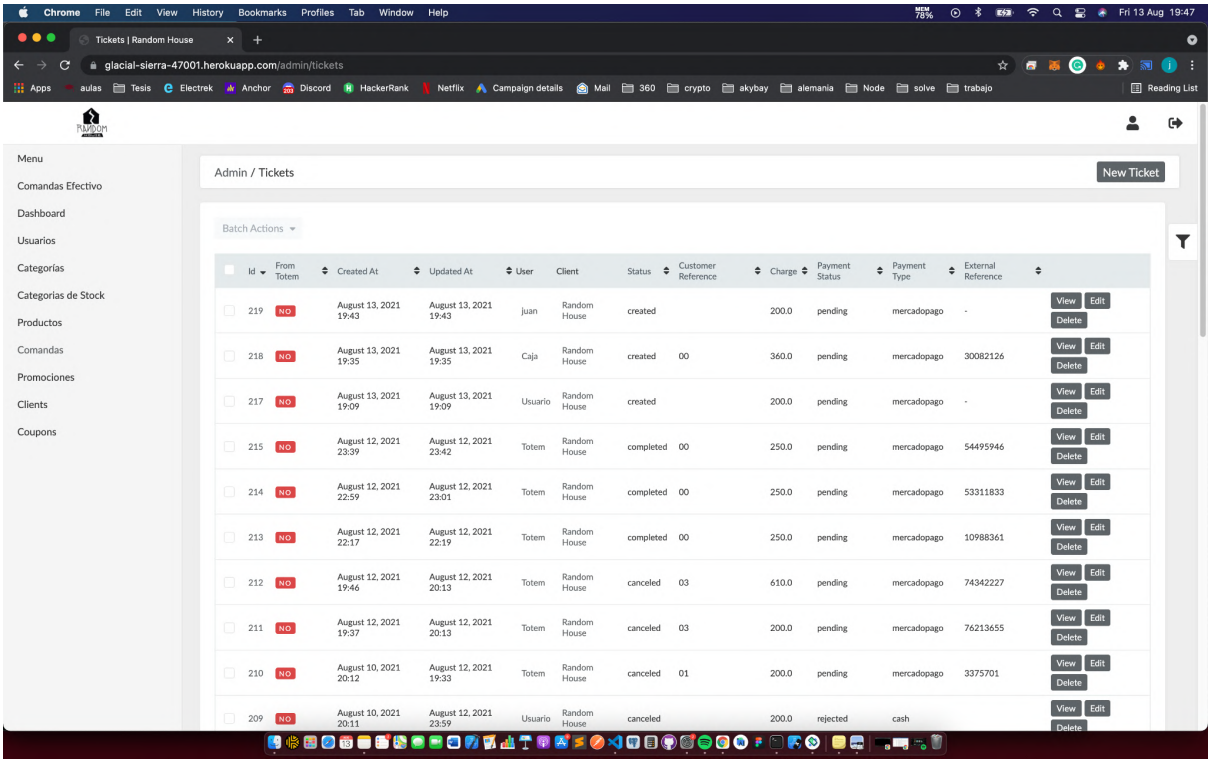


Figura 42: Tickets Backoffice

2.6.6.8. Comandas efectivo

Aquí se encuentra el listado de todas las comandas recibidas por el sistema que fueron pagadas en efectivo.

2.6.6.9. Promociones

Aquí se encuentra el listado de promociones, en donde se pueden agregar, editar y eliminar las mismas. También se ofrece la opción de que se pueda activar/desactivar una promoción y la opción de poder diferenciar entre promoción destacada y no destacada.

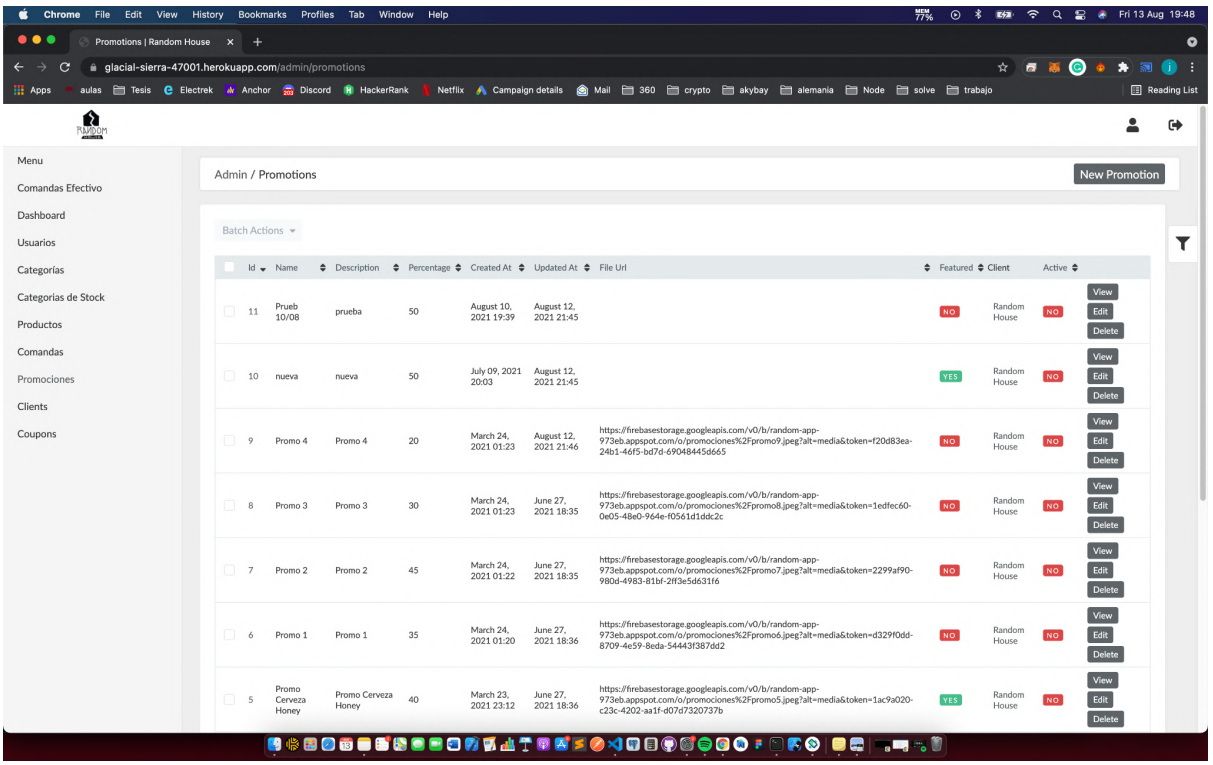


Figura 43: Promociones Backoffice

2.6.6.10. Cupones

Aquí se encuentra un listado de los cupones generados por los administradores. De esta manera se pueden visualizar cuales se encuentran activos y cuáles no, siendo activos los que aún no han sido consumidos e inactivos lo que ya fueron consumidos. También pueden crear nuevos cupones, el flujo para crear los mismos es el siguiente:

Debe elegir el nombre del código (ej. SÁBADO15%), luego debe elegir el porcentaje de descuento que tendrá ese cupón y por último deberá elegir a los usuarios que desea enviar el mismo. En caso de no elegir a ningún usuario el cupón será válido para todos.

Un ejemplo de caso de uso de las dos formas de asignar los cupones serían los siguientes:

Si el administrador desea resarcir a un cliente en específico por algún problema que tuvo, por ejemplo, se demoró la comida o lo que le entregaron no fue lo que el cliente pidió, le podrá asignar el cupón con el descuento que desea a esa persona en específico.

Si por ejemplo el administrador sabe que el sábado habrá mal tiempo y que eso afecta directamente en la cantidad de clientes que se acercan al bar, podrá enviar un cupón a todos ellos con el descuento que desee, de forma tal de incitar a los clientes a ir al bar.

Cuando se crea un cupón se envía una *push notification* a todos los usuarios que han sido seleccionados.

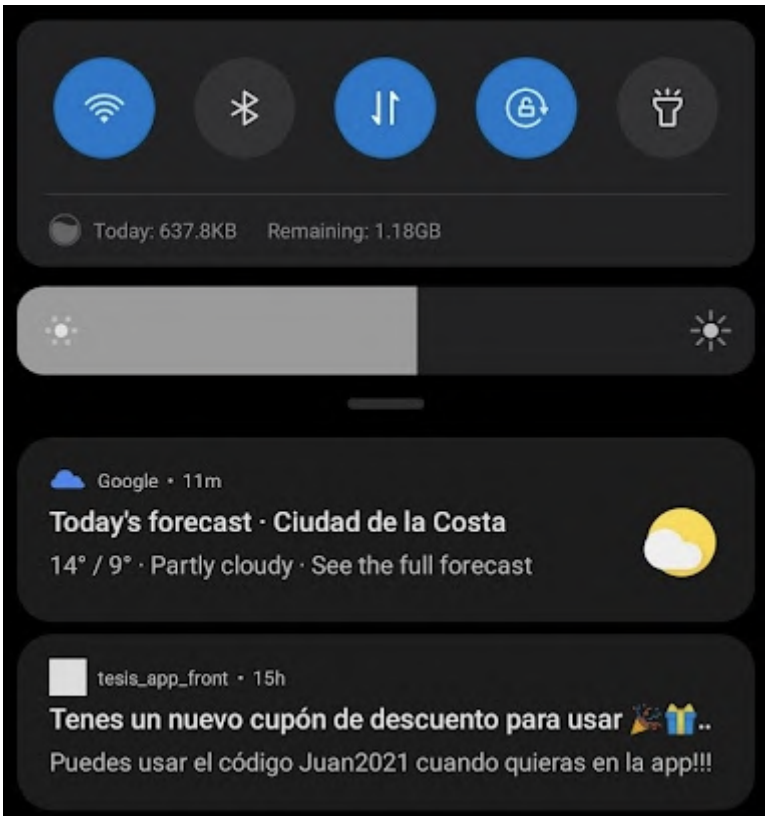


Figura 44: Notificación cupón

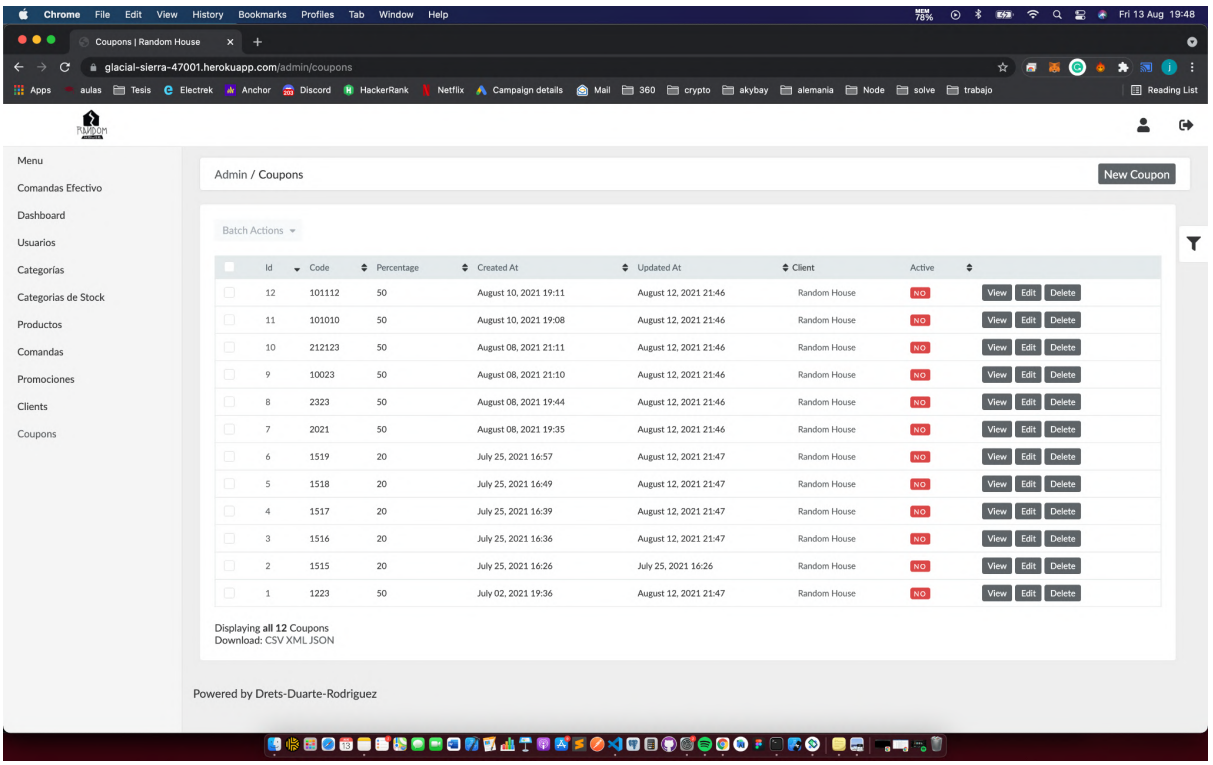


Figura 45: Cupones Backoffice

2.6.7. Usuarios del sistema

- Usuario de terminal autoservicio
- Usuario de aplicación mobile (iOS/Android)
- Usuario de comandas vía web (cajero)
- Cocina
- Usuario administrador del bar (*back office*)

3. Enfoque de trabajo

En el presente capítulo se describen las características del proyecto y el equipo de las cuáles se parte para optar por las herramientas y metodologías adecuadas para su desarrollo.

3.1. Características del proyecto

El proyecto desde un principio contó con el compromiso del cliente para participar durante el desarrollo del mismo. El cliente ya tenía una idea clara de lo que buscaba resolver en su modelo de negocio. A su vez, por la naturaleza del producto, es un proyecto que va a cambiar durante el proceso dependiendo fuertemente de los usuarios y sus experiencias.

Los usuarios cumplieron un papel muy importante para la definición de las formas de trabajo, es un proyecto el cual consiste en cambiar la dinámica de atención que tiene el bar y a la cual están acostumbrados sus clientes. Por lo que desde el inicio hasta el final del desarrollo se supo que su participación iba a ser constante y se les debería tener especial consideración.

3.2. Características del equipo

El equipo se encuentra conformado por tres estudiantes de los cuales dos han trabajado previamente juntos en tareas de la universidad en algunas materias, por lo que un integrante del equipo era nuevo.

En cuanto a las tecnologías, *Ruby on Rails* era completamente nuevo para dos integrantes y el tercero solamente la había usado en una materia. *Flutter* también era desconocido para dos integrantes del equipo y el tercero ya tenía un poco de experiencia con la tecnología.

Todos los integrantes trabajan además de estudiar lo que reduce las horas disponibles, además del trabajo dos integrantes tuvieron que cursar materias y dar exámenes en el correr del año de trabajo del proyecto.

3.3. Ciclo de vida

Dadas las características mencionadas el equipo optó por la utilización de un ciclo de vida iterativo e incremental.

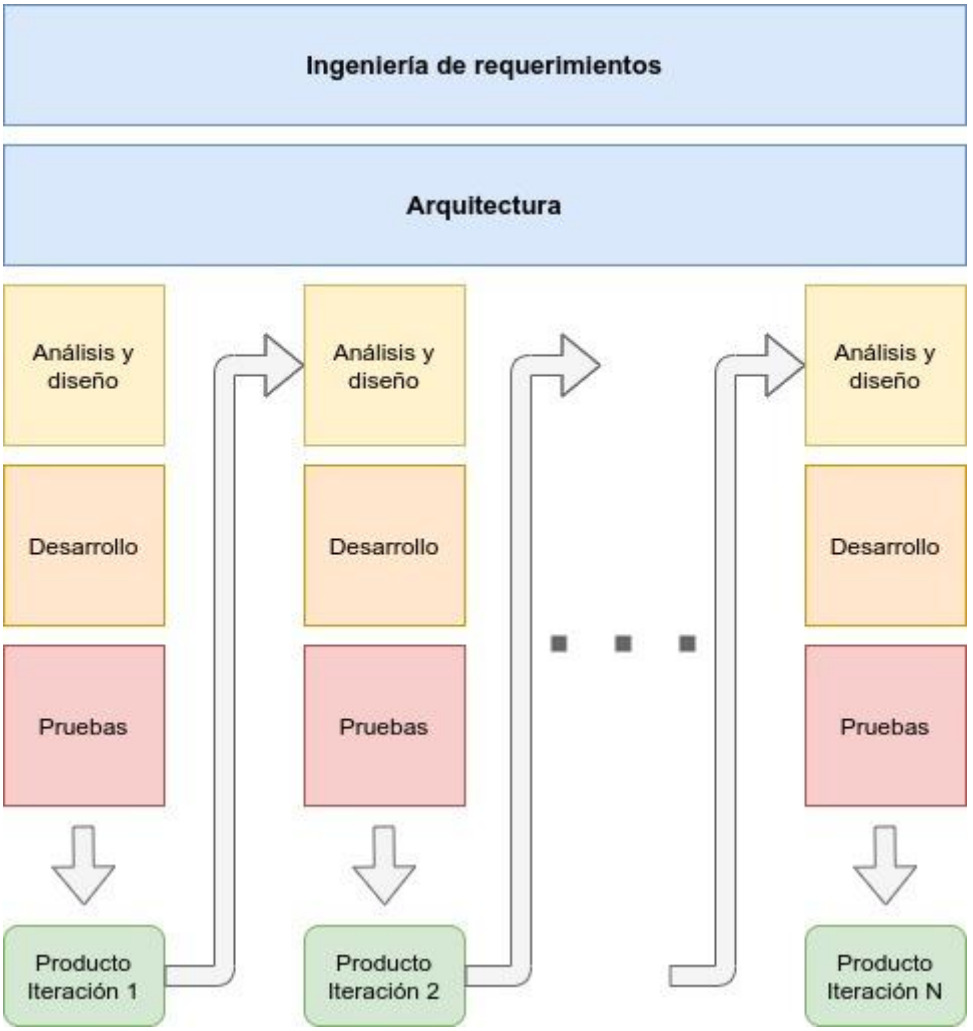


Figura 46: Ciclo de vida

3.3.1. Justificación de elección

Se decidió que el ciclo de vida iterativo e incremental era el correcto para llevar adelante el proceso y realización de tareas del proyecto por los siguientes motivos.

El cliente contaba con las funcionalidades principales para el desarrollo del sistema, por lo que había una base de donde arrancar y definir una arquitectura inicial. Luego sobre esto se realizan múltiples iteraciones del proceso de software permitiendo obtener continuo *feedback* tanto del cliente como de los usuarios.

De la primera iteración se desprende una primera versión inicial (Mínimo Producto Viable). La idea es probarla con los usuarios, evaluar su uso, recibir *feedback*, refinarla y generar “n” versiones hasta que se desarrolle el sistema adecuado. Dadas las características del proyecto es necesario contar con una rápida realimentación del usuario y en cada iteración del ciclo ejecutar actividades de análisis, desarrollo y pruebas.

La descripción del proyecto demostró que obtener la totalidad de los requerimientos desde el comienzo iba a ser una tarea extremadamente difícil. Si bien el cliente tenía bien claro cuál era el problema actual del bar, para poder llegar a todos los requerimientos funcionales del sistema, se iba a necesitar pasar por diferentes iteraciones.

A lo anterior se le suma que los requerimientos iban a cambiar durante la completitud del proyecto. Se busca centralizar todo el flujo de las órdenes, desde que el cliente realiza la compra, se hace el pago, la comanda llega a la cocina y se descuenta del *stock*. Por lo que muchas áreas se involucran en el proyecto, llevando a la posibilidad de que varias complicaciones se presenten a la hora de su desarrollo. Como se busca atacar muchos frentes del negocio a la vez, cada uno de estos sectores puede solicitar modificaciones o nuevas solicitudes. Además, estos cambios pueden influir en los requerimientos del otro.

Otro punto importante que muchas veces no se toma en cuenta es que el cliente no sabe de informática, causando posibles problemas en la comunicación.

Todo lo anterior llevó al equipo a pensar que lo mejor es realizar un *software* que pueda ser evaluado en cada iteración y que se le vayan incorporando cambios necesarios de manera de incrementar el valor.

3.3.2. Fases del ciclo de vida

Ingeniería de requerimientos

En esta etapa el objetivo es identificar, especificar y validar los requerimientos del sistema.

Al comenzar con el proyecto se recabaron los requerimientos partiendo de las

necesidades conocidas por el cliente, encuestas, entrevistas, observación e *ingeniería inversa*.

Todo lo obtenido es especificado en forma de historias de usuario y se incorpora en el *Product Backlog* inicial.

Arquitectura

Etapa en la cual se diseñó una primera solución arquitectónica cumpliendo con los requerimientos funcionales y no funcionales recabados. En esta solución ya se incluyeron las tecnologías adecuadas para el desarrollo del sistema.

Análisis y diseño

En cada iteración es necesario pasar por el proceso de Ingeniería de Requerimientos nuevamente para poder determinar correctamente que nuevas funcionalidades incluir.

Se pasa por procesos de validación en conjunto con el cliente y usuarios donde se obtiene todo el *feedback* necesario sobre el producto hasta el momento.

Es importante observar a los potenciales clientes del bar usando las distintas interfaces para comprender qué impacto genera en ellos las nuevas actualizaciones.

De lo anterior se desprenden nuevos cambios y funcionalidades que se necesitan para mejorar e incrementar el sistema que van a ser incluidos en el *Product Backlog*. También en esta etapa es posible que sea necesario hacer cambios sobre la arquitectura del sistema.

Otro aspecto importante a mencionar es que para cada nuevo requerimiento que involucre UI se crean *mockups* rápidamente, facilitando el intercambio con el cliente quien lo puede visualizar y validar antes de su desarrollo.

Desarrollo

Se tiene como objetivo el desarrollo de los requerimientos definidos y seleccionados en la etapa de análisis y diseño. La idea es trabajar iterativamente sobre la versión anterior e incrementar su valor.

El resultado de esta fase es el código fuente de una nueva versión pronta para ser instalada y probada por el cliente y potenciales usuarios.

Pruebas

En esta etapa se realizan las pruebas de *software* necesarias para validar su correcto funcionamiento, luego se despliega una nueva versión del producto.

3.4. Procesos de Gestión

3.4.1. Metodologías de trabajo

Dadas las características del proyecto, el equipo optó por la utilización de una metodología ágil, más específicamente el *framework Scrum* para su gestión, pero no en su estado puro, se explicará su adaptación más adelante.

El *framework* propone entregas incrementales que se adecuan muy bien a las exigencias del proyecto.

Lo mencionado no es la única motivación que llevó a su selección, para el equipo esto forma parte del desafío y aprendizaje, el continuar con lo aprendido en la carrera de grado. Ninguno de los integrantes anteriormente trabajó con metodologías ágiles, esto involucró investigación y un arduo trabajo para poner en práctica las herramientas que la metodología provee.

Por otro lado, adecuado a las características del proyecto, se determinó como un factor clave las continuas iteraciones para modificar y mejorar la solución de acuerdo al *feedback* obtenido. *Scrum* ejecuta el proyecto en fases cortas de dos, o como mucho, cuatro semanas, permitiendo mucha flexibilidad ya que en cada una de estas fases se replantean tareas y objetivos. Haciendo énfasis en este punto, el *software* funcional es la primera medida de progreso según el manifiesto ágil, permitiendo hacer entregas tempranas y sus validaciones.

Otro punto a destacar es la falta de exigencia por parte del cliente de documentación, si bien se realiza documentación en el comienzo y las distintas etapas, el cliente no exige una documentación extensa.

Mencionado lo anterior, es necesario aclarar que se utilizan algunos elementos de las metodologías tradicionales, como lo son la gestión de riesgos, responsabilidades de ingeniería de *software* dentro del equipo y la realización de un plan y aseguramiento de calidad.

De los eventos propuestos por la metodología *Scrum*, se utilizan: *daily meetings*, *sprint planning*, *sprint retrospectives* y *sprint reviews*. Dentro de estos se realizaron modificaciones como por ejemplo no realizar *daily meetings* todos los días y que el *product owner* era un integrante del equipo. Las tareas se distribuyen en *sprints* de dos semanas, en el caso de no poder culminar con una iteración, o si surgen incidentes, estos se asignan a la próxima iteración. Para definir un criterio único de asignación de *story points* a las historias de usuarios, al principio del proyecto el equipo seleccionó una historia de usuario muy sencilla y el equipo se puso de acuerdo en un valor a ser asignado, a partir de ese valor se asignaron los *story points* del resto.

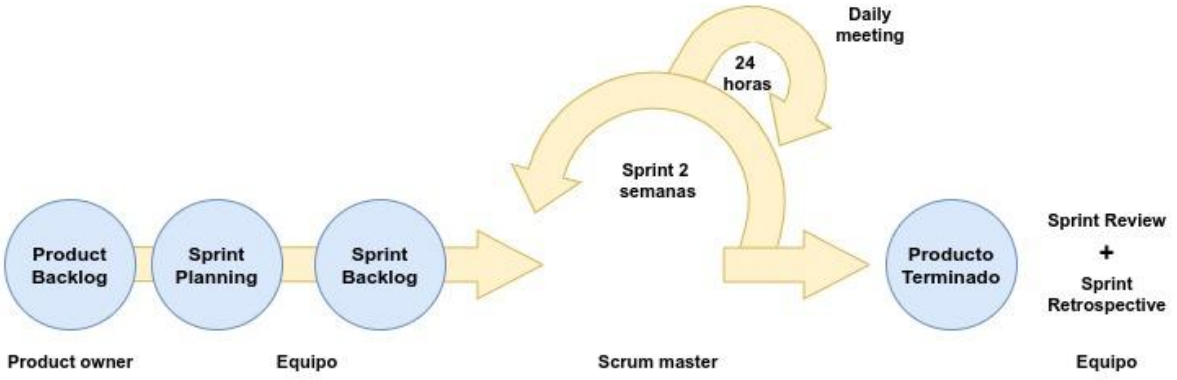


Figura 47: Scrum

3.4.2. Aplicación de Scrum

Scrum se basa en el aprendizaje continuo y en la adaptación a los cambios constantes, reconoce que el equipo no lo sabe todo al inicio del proyecto y evoluciona a través de la experiencia. Para su aplicación el marco *Scrum* cuenta con las siguientes herramientas.

3.4.2.1. Artefactos

Cuando se habla de “artefactos” se refiere a una herramienta para solucionar un problema. En *Scrum* se pueden encontrar artefactos, tales como, el *product backlog*, *sprint backlog*, *burndown charts* y el producto terminado o producto interno definido como un “hecho” al final de cada *sprint*.

Figura 48: Artefactos [8]

Product Backlog

En simples palabras, se trata de la lista de tareas por hacer del equipo. Es una lista dinámica donde se incluyen funciones, requisitos, mejoras, correcciones que luego actuarán como entrada para el *sprint backlog*. El *Product Owner* (propietario del producto) es el encargado de constantemente hacer una revisión, cambio de prioridades y mantenimiento de este *backlog*. Continuando con la idea planteada en secciones anteriores, al evolucionar el producto sabemos que los requerimientos podrían sufrir cambios, surgir nuevos o algunos dejar de ser relevantes, por lo que es necesario mantener actualizado el *product backlog*.

Sprint Backlog

Esta se trata de una lista de historia de usuarios o correcciones de incidentes seleccionadas por el equipo para desarrollar antes del inicio de un nuevo *sprint*.

Esto se hace en la *Sprint* Planning (explicado en 3.4.2.2. Eventos), donde el equipo se reúne y elige qué tareas incluir. El *sprint backlog* puede ser flexible y evolucionar durante la iteración, pero nunca puede perder el objetivo actual que el equipo busca en el *sprint*.

Producto terminado

Es el objetivo del *sprint*, corresponde a un producto final funcional luego de terminar un *sprint*, donde se puede visualizar el objetivo actual logrado. Se deben evaluar las tareas realizadas para determinarlas como aceptadas dentro de lo requerido según las historias de usuario o incidentes elegidos para desarrollar.

3.4.2.2. Eventos

De forma periódica dentro del *framework Scrum* se realizan un conjunto de eventos o reuniones. La forma de realizarlos puede variar de acuerdo al contexto del proyecto y del equipo, aquí se plantean y detalla su forma de uso.

Daily meetings

Trata de una reunión diaria de muy corta duración que se realiza siempre a la misma hora, normalmente se llevan a cabo por las mañanas antes de comenzar con el desarrollo y se intenta que duren menos de 15 minutos.

Al tratarse de un proyecto de grado los integrantes del equipo durante el día debían cumplir con responsabilidades laborales, por lo que se decidió que lo más conveniente para no detener el dinamismo y velocidad de trabajo sería realizarlas por mensajería instantánea, como WhatsApp.

El objetivo de esta reunión es que todo el equipo esté en sintonía de acuerdo con los desafíos del *sprint* y la situación de cada uno, así pudiendo planificar el trabajo del día. El medio seleccionado se adecúa perfectamente al contexto de trabajo y cumplió con las expectativas esperadas en las *daily meetings*.

Sprint planning

Durante esta reunión todo el equipo planifica que se va a desarrollar durante el *sprint* actual, se define cual va a ser el objetivo durante las próximas dos semanas.

Posteriormente, se evalúa que *user stories* del *product backlog* se van a incluir en el *sprint*. Estas historias siempre se adecuan al objetivo y las prioridades asignadas, además debe ser factible su implementación durante la iteración teniendo en cuenta posibles cambios en la velocidad del equipo. Este punto no es menos importante, ya que al tratarse de tecnologías nuevas para el equipo no se puede contar con la misma velocidad al iniciar comparado con etapas avanzadas del proyecto. Por la misma razón, en esta reunión se hace un replanteo de los *stories points* asignados a cada *user story*, es posible que algunas estimaciones no hayan sido las más correctas y luego con más conocimiento sobre las tecnologías el equipo puede ser capaz de mejorarlas.

Al concluir la reunión, cada integrante debe tener claro qué y cómo se va a entregar el próximo producto terminado.

Previo a la realización de estas reuniones el *Product Owner* se encargaba de tener el conocimiento suficiente para ser capaz de priorizar los requerimientos e incrementar el producto de acuerdo a lo que el cliente requería. Eso lo realizaba en un comienzo con reuniones presenciales con las dueñas del bar, luego debido al Coronavirus estás pasaron a ser por distintos medios, pudiendo ser llamadas por Google Meets o telefónicas.

Sprint reviews

Al final de cada *sprint* el equipo se reúne para mostrar lo realizado en él. Cada integrante muestra las *user stories* terminadas y se recibe comentarios sobre el trabajo realizado. El *Product Owner* decide si se llegó a lo esperado para liberar un incremento.

Por practicidad, estás se realizan el mismo día que la *sprint planning* y se intenta que entre las dos reuniones no se tarde más de dos horas.

Sprint retrospectives

Aquí el equipo se reúne para analizar qué está funcionando y que no funciona en el *sprint*. De esto se puede deducir que se debe mejorar o cambiar en los procesos o

herramientas. Es el momento para discutir sobre lo que no está bien y revisar soluciones.

Debido al Coronavirus y la forma en que esto afectó a la presencialidad y comunicaciones por momentos, esta etapa fue importante ya que era necesario tener un lugar donde evaluar cómo continuar con algunas formas de trabajo.

3.4.2.3. Roles

El equipo se *Scrum* se compone de tres roles específicos: el *Product Owner*, el *Scrum Master* y el equipo de desarrollo.

El equipo de desarrollo es multidisciplinario, esto quiere decir que está compuesto por programadores, encargados de SQA [47], *SCM*, UX, etc.

Aplicado al proyecto cada integrante del equipo va a ser asignado a uno de los roles de *Scrum* y las responsabilidades de las áreas mencionadas van a ser distribuidas.

Product Owner (Rodrigo Duarte)

El *product owner* es quien más conoce del producto. Su principal tarea es entender los requisitos del cliente, su negocio y el mercado para luego ser capaz de priorizar el trabajo que el equipo va a realizar. Su visión debe ser la de dar valor tanto al equipo como al producto en proceso.

Dentro de sus tareas se encuentra: gestionar el *product backlog*. Comprender y comunicarse con el cliente y equipo para asegurarse de que todos estén bien direccionados. Establecer directivas claras sobre los próximos desarrollos. Tomar decisiones sobre la correctitud del trabajo hecho.

De lo conversado con el cliente, desde un principio se decidió que el rol de *product owner* lo tome un integrante del equipo. Las dueñas del bar estuvieron a disposición para trabajar en conjunto al tomar decisiones de requerimientos, evaluaciones, priorizaciones, pero al comentarles las tareas que el *product owner* debe realizar fue de su preferencia tomar esa decisión y ellas trabajar constantemente con esa persona para que pueda cumplir con sus responsabilidades.

Scrum Master (Juan Drets)

Es la persona con más conocimientos sobre las prácticas de *Scrum*. Debe proporcionar constantemente conocimientos a todo el equipo sobre las distintas reglas y cerciorarse de su correcto uso. Debe encargarse que se utilice el *framework* correctamente para mantener un flujo dinámico, planificando bien los distintos recursos.

Equipo de desarrollo

El equipo de desarrollo es el que saca el trabajo adelante. Se encarga de impulsar el plan de cada *sprint*. Deciden cuánto trabajo creen que pueden realizar en una iteración según su historial de velocidad. Mantener *sprints* de longitud fija ayuda al equipo de desarrollo a recibir *feedback* sobre la estimación y la entrega concluida, lo que hace que los planes sean cada vez más precisos.

Se definieron distintas responsabilidades de ingeniería de *software* y estas fueron asignadas a cada uno de los integrantes dentro del equipo de desarrollo.

A pesar de esta división, todos los miembros del equipo se ayudan entre sí para asegurar la finalización satisfactoria del *sprint.*

Gerente de proyecto	Juan Drets
Responsable de SQA	Juan Drets y Sebastian Rodriguez
Analista de requerimientos	Sebastian Rodriguez
Arquitecto	Rodrigo Duarte
Responsable de SCM	Rodrigo Duarte
Desarrollador	Todos
Tester	Todos

Figura 49: Roles

Gerente de proyecto

Encargado de dirigir al equipo durante el proyecto, cumple con el rol de líder. Debe estar en caso de problemas y ser una guía para el equipo y sus funciones. Está interesado en la calidad de los resultados por lo que su objetivo es garantizar que el proyecto se lleve a cabo de la mejor manera posible.

Responsable de SQA

Es responsable de asegurar la calidad de los productos generados a lo largo del proyecto. Para esto debe planificar y documentar actividades de SQA y definir estándares de productos.

Asimismo, debe asegurarse de que se defina y utilice un proceso de *software* en el proyecto, apoyando al equipo en su definición y documentación. Durante el proyecto, puede evaluar la eficacia del proceso y mejorarlo en caso de ser necesario.

Analista de requerimientos

Encargado de que el proceso de ingeniería de requerimientos se realice de manera adecuada. Su función es recolectar y validar la información del cliente, para luego especificarla de la manera definida. Cumple un rol importante ya que se encuentra involucrado en la definición del producto al que se busca llegar.

Arquitecto

Es el responsable de tomar las decisiones técnicas para la construcción de la solución tecnológica. Apoya al analista en la construcción de los requerimientos del sistema y el diseño para alcanzar un producto de calidad cumpliendo con los principios de diseño y arquitectura.

Responsable de SCM

Es el encargado de proporcionar la infraestructura y entorno para la Gestión de Configuración a lo largo del ciclo de vida del proyecto. Es un soporte para la actividad de desarrollo para que el equipo cuente con los ambientes apropiados para realizar y verificar su trabajo.

4. Ingeniería de requerimientos

4.1. Introducción

En la siguiente sección se describen los procesos de ingeniería de requerimientos realizados. Esto abarca actividades de relevamiento de funcionalidades, su especificación y validaciones.

Como se ha mencionado anteriormente, el cliente ya tenía una idea establecida sobre una base de requerimientos por donde comenzar. Dado esto, se estableció una primera instancia para especificar estos requerimientos y validar su correctitud de acuerdo a sus objetivos y continuar con el mínimo producto viable.

4.2. Proceso

El proceso se basa en comprender las necesidades del cliente y la de sus usuarios, siempre teniendo especial foco en estar trabajando sobre el camino correcto. Para asegurar esto, el proceso se realiza reiteradas veces durante el desarrollo del proyecto, en cada iteración del ciclo de vida.

Consiste en dos etapas que se realizan en secuencia, en caso de ser necesario se puede reiterar la secuencia hasta concluir en los artefactos correctos.

La primera etapa está compuesta por dos actividades, la extracción y análisis de requerimientos y la especificación. Aquí se hacen tareas como encuestas, entrevistas, reuniones con usuarios para lograr identificar y especificar los requerimientos. Antes de terminar con el proceso se pasa por la segunda etapa, la validación, donde en conjunto con el cliente se hace una revisión de lo recabado para poder validar que todo esté en línea con los objetivos del proyecto.

Finalmente, se desprende lo trabajado en la especificación de requerimientos funcionales y no funcionales.

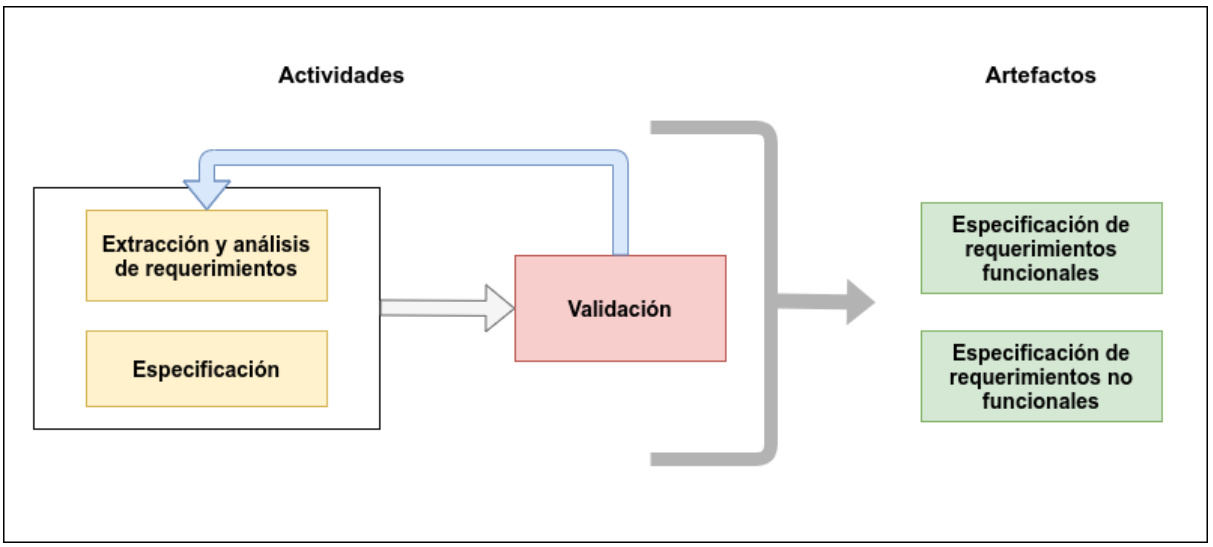


Figura 50: Proceso

4.3. Metodología de trabajo

Como se encuentra mencionado, el cliente presentó una lista con funcionalidades base para la solución que pretendían abordar. Éste fue el punto de partida por el que se comenzó a trabajar y se tuvo en cuenta para definir los primeros requerimientos. De igual manera se consideró necesario una etapa inicial de evaluación de estos requerimientos para así poder validar que realmente se adecue a las necesidades de sus usuarios y aporte significativamente una mejora en su modelo de negocio.

Luego de que Random House aceptara esta iniciativa y estuvieran dispuestos a que surjan cambios en su planteo, se comenzó investigando e identificando algunos actores principales para hacerles entrevistas, estas entrevistas se pueden ver en el Anexo 12.5. Entrevistas. A la vez se dispuso una serie de preguntas en encuestas para que respondan potenciales clientes del bar.

De esta fase de estructuración y organización se realizaron algunos cambios en los requerimientos planteados, por lo tanto, fue necesario presentarlos al cliente y que validen si estaban de acuerdo con ellos. De esta etapa se concluyó en un *product backlog* priorizado de manera adecuada para realizar el primer *MVP*. Esta etapa se le denominó *sprint* 0 la cual corresponde a la fase previa al inicio del proyecto de *Scrum* y su objetivo fue establecer el propósito del proyecto, así como las líneas principales de trabajo.

Además de lo mencionado, otras de las actividades realizadas en esta etapa fueron: el estudio del diseño y la arquitectura del sistema, la asignación de roles de cada uno de los integrantes organizando el trabajo a realizar, se comienza a analizar plazos y esfuerzos necesarios para culminar en fecha los objetivos.

Aunque el *sprint* 0 no existe oficialmente en el *framework Scrum*, el equipo la consideró crucial para tener claras las bases del proyecto y la metodología ágil que se usó facilitando su aplicación.

4.3.1. Extracción y análisis de requerimientos

Diversas técnicas y herramientas fueron realizadas durante la extracción y análisis de requerimientos durante todos los ciclos de vida. Se presentan a continuación.

4.3.1.1. Encuestas

Metodología y motivación

El equipo realizó encuestas teniendo como objetivo la segmentación realizada de los potenciales usuarios del bar, estas se pueden ver en Anexo 12.3. Encuestas. Para ello se construyó una encuesta en Google Forms y se distribuyó en las redes. En el principio de la encuesta se consultó la periodicidad a la que se concurre a bares cerveceros para así poder filtrar y extraer respuestas más exactas a nuestras necesidades.

Luego, en la encuesta se busca responder a preguntas como: ¿Qué le parece el servicio generalmente en estos bares? ¿Se siente cómodo realizando una orden a través de una pantalla (celular o tótem)? Con estas se buscó evaluar la viabilidad de la solución, se necesitaba saber si los usuarios iban a estar de acuerdo con el cambio en el modelo de atención e iban a querer usarlo.

Resultados

En los resultados se puede ver que la segmentación fue aplicada correctamente, más del 90% de las respuestas fueron de personas que concurren más de tres veces al mes a bares. El 75,3% de las respuestas concuerda que realmente hay un problema en la velocidad de atención en horarios concurridos. Siguiendo esto, se

consultó si estarían cómodos haciendo el pedido a través de un tótem de autoservicio, a lo cual el 85% respondieron que sí. En modo de obtener respuestas abiertas se realizaron preguntas concretas sobre el valor que cree el encuestado que se estaría aportando, de ellas el factor común que se obtiene es la velocidad en la atención. Luego se consulta sobre incluir juegos interactivos en la aplicación, esto no fue tan impulsado como las demás funcionalidades, menos del 60% cree que es una buena idea.

En este punto comenzamos a creer que se debe hacer énfasis en mejorar la velocidad del flujo de pedido de una orden y reevaluar algunos de los requerimientos propuestos.

4.3.1.2. Entrevistas

Metodología y motivación

Se realizaron dos tipos de entrevistas enfocadas a dos de los actores claves identificados, a los funcionarios y a los clientes del bar. De los funcionarios se eligió a cocineros y cajeros.

Debido a la situación de pandemia no se pudo concurrir con la frecuencia que se hubiese querido, pero de todas formas en algunas ocasiones se concurrió presencialmente al local. En el resto de las ocasiones se debieron hacer de forma virtual causando que se pudiera perder información no verbal transmitida por los entrevistados. Las mismas se hicieron a través de plataformas como Google Meets, Zoom y llamadas telefónicas.

El objetivo de las entrevistas a funcionarios fue conocer a detalle en qué consiste su trabajo y comprender desde su punto de vista el flujo de trabajo que tienen. Por otro lado, de los clientes se buscó entender su opinión sobre la atención del bar y conocer qué piensan sobre algunas de las funcionalidades del sistema.

En ambos tipos de entrevistas se plantearon preguntas específicas con el fin de validar requerimientos y algunas más amplias para obtener pensamientos respecto a algunos temas y encontrar posibles problemas no identificados.

Procedimiento

Se buscó que las entrevistas sean lo más productivas posibles, por lo que previamente se realizaron esqueletos con preguntas ya pensadas para llevar un hilo conductor coherente.

Las entrevistas naturalmente se desvían un poco del esquema planeado, por lo que en todas se definió un interlocutor encargado de que no se generen problemas en la comunicación. También al menos un integrante más debía estar presente para poder tomar nota de las respuestas y observaciones pertinentes.

Cada entrevista comenzaba con una introducción presentándonos ante el entrevistado, contando un poco el contexto del proyecto, buscando generar un clima de confianza. Luego el proceso de preguntas varía si se trata de clientes o funcionarios.

Para los clientes se comienza preguntando qué esperan de la aplicación sin antes detallar sus principales funcionalidades, lo que nos permite encontrar qué puntos de dolor perciben. Luego se les pasa a detallar cómo funciona la solución y se le hacen preguntas específicas para validar ideas. Dependiendo del momento en que se realiza la entrevista se les presenta un prototipo sobre el cuál se les pide su opinión y se analizan posibles problemas de usabilidad.

Por otro lado, a los funcionarios primero se les consulta respecto al modelo de negocio actual y que piensan al respecto. Luego se les explica la solución, se les presenta un prototipo y se les consulta qué les parece respecto al trabajo que hacen.

Resultados

De las entrevistas se pudo terminar de comprender muchos de los problemas más a profundidad y encontrar nuevos puntos de dolor que aportaron nuevas funcionalidades a tener en cuenta en la solución.

Se comprendió en su completitud el flujo de atención que están utilizando actualmente donde se pudo identificar algunos puntos importantes que debían ser mejorados. Para hacer un pedido es necesario enviar un WhatsApp al bar con los

productos que se desean consumir. Pudimos detectar problemas tanto del lado de los funcionarios como de los clientes. El cliente a la hora de realizar la orden se le hacía muy dificultoso ingresar los productos de memoria en un mensaje de WhatsApp. Por el lado de los funcionarios, el cajero es el encargado de recibir los mensajes de los pedidos y pasarlos al sistema para ingresarlos como una comanda. Esto es una tarea muy engorrosa y por momentos muy difícil de gestionar ya que los mensajes llegan todos a una cuenta donde van surgiendo nuevos mensajes, tanto de órdenes como consultas, comenzando a ser una actividad muy confusa y hasta casi imposible de manejar.

Luego de recibido el pedido e ingresado al sistema se genera una comanda que debería ser recibida por la cocina. El bar no cuenta con una forma automática de hacer esto, por lo que una persona debe imprimir un *ticket* y alcanzarlo al cocinero. Esto puede parecer que no es un problema, pero luego en momentos picos donde el cajero tiene que atender muchos clientes se comienza a transformar en un caos donde no hay trazabilidad real de las comandas solicitadas.

Por otro lado, de entrevistas con clientes se pudo deducir que no era de gran aporte a la solución incluir juegos en la aplicación. Este es un punto que se visualiza en las encuestas y es reforzado en las entrevistas. Los clientes en la mayoría de los casos terminan valorando mucho más que se incluyan promociones, cupones, elementos que le generen descuentos en la compra y no tanto formas de entretenerse digitalmente en el bar.

En primera instancia nuestro proyecto iba a ser una aplicación *mobile* en donde el usuario iba a poder realizar sus pedidos a través de la misma y además iba a incorporar una serie de juegos simples como por ejemplo trivias en donde los usuarios iban a poder jugar mientras esperaban su pedido. La idea era también que estos juegos puedan ser jugados entre los usuarios que se encontraban en el bar de forma que los usuarios puedan interactuar entre ellos, aunque sean desconocidos.

Una vez terminado el *sprint* 0 pudimos concluir que al cliente le aportaría mucho más valor entregarle una solución integral en donde se procesen todos los pedidos por este sistema y que esa información se pueda utilizar para entregar gráficas con información relevante para el cliente además de un control de *stock* centralizado y lo

más preciso posible ya que hasta el momento lo realizaban una vez por semana y en papel. Es por esto que el equipo decidió enfocarse en desarrollar un sistema de toma de comandas que cubra todas las posibles formas de realizar pedidos en conjunto con un sistema de gestión de comandas para la cocina y un panel administrativo para gestionar la carta, los cupones, las promociones y el *stock*. Dado que se realizó este cambio luego del *sprint* 0 se evaluó el alcance del mismo y seguía siendo adecuado para el tiempo de trabajo y la cantidad de integrantes que conforman al equipo.

4.3.1.3. Observación

Metodología y motivación

Está fue una técnica muy importante durante todos los ciclos de vida realizados.

En la primera etapa de donde se recabaron requerimientos con esta técnica se obtuvo información de primera mano sobre las actividades que se efectúan en el bar. Esté método permite observar cómo se llevan a cabo los procesos y verificar que todo sea haga de acuerdo a lo transmitido por parte del cliente. Muchas veces lo que sucede en la realidad no es exactamente lo que se especifica y por medio de la observación esto puede ser validado. También es una muy buena manera de identificar puntos de dolor desde una mirada más técnica buscando la optimización del negocio.

Ya conociendo el negocio, en los siguientes ciclos de vida, se continuó utilizando la observación con varios aspectos especificados previamente para analizar.

Procedimiento

Antes de llevar adelante la técnica de observación el equipo se preparaba para aprovechar de mejor manera está instancia.

El equipo determinaba la información necesaria a la que se debía tener especial atención en el momento de la observación. Por ejemplo, qué funcionalidades nuevas deben ser evaluadas, qué momento del proceso se vio afectado por nuevos cambios. Luego se determinaba el día y horario para la visita al bar, ya que dependiendo de lo que se quiere observar se debía asistir en horarios normales o

más con más concurrencia. Después de tener esto definido se le informa al cliente en qué momento el equipo iba a estar trabajando en el bar. Por último, se planifica de qué manera se hacen las anotaciones, para esto cada integrante del equipo tomaba sus apuntes y luego en otra instancia se analizan y se sacan conclusiones.

Resultados

Mediante la observación fue que se pudo terminar de comprender algunos aspectos específicos del flujo de pedidos, como algunos problemas dependiendo del horario y día que no son mencionados en la generalidad del proceso.

Se pudo observar que el cliente realizaba de manera manual en una planilla impresa la actualización del *stock* según las ventas realizadas. De esto se le propone al cliente incluir el control de *stock* dentro de la solución ya que se consideró como una funcionalidad que realmente les solucionaría un problema y le liberaría mucho tiempo para concentrarse en otras tareas.

En las primeras versiones de la solución el cliente cada vez que realizaba un pedido debía hacer el pago para finalizarlo. Se pudo ver que muchos de los clientes tenían la necesidad de seguir ordenando de manera fluida y hacer el pago al momento de retirarse.

Por otro lado, pudo validarse que en horarios picos se generaba un efecto embudo de atención en la caja al recibir muchos pedidos y no contar con una forma automática que optimice el proceso.

4.3.1.4. Ingeniería inversa

Metodología y motivación

Como parte de la extracción de requerimientos se llevó adelante el proceso de *ingeniería inversa*. Se busca estudiar otras soluciones similares en el mercado con el objetivo de visualizar qué funciones incluyen y de qué manera lo resuelven.

Como parte de la investigación se incluyeron soluciones similares a la nuestra. Ellas son:

- Menoo [48]: aplicación uruguaya en donde se puede realizar reserva de mesas y pedidos desde la app.
- SoftRestaurant [49]: sistema de gestión para restaurantes que además tiene la opción de pedidos por medio de una terminal autoservicio. Dicha terminal es difícil de utilizar y presenta problemas en los usuarios a la hora de realizar pedidos.
- DeliSoft: es un sistema de gestión para restaurantes al igual que SoftRestaurant, particularmente es el sistema que es utilizado en Random House.
- PedidosYa [50]: aplicación para realizar pedidos tanto a restaurantes como a otro tipo de tiendas.

Resultados

Pudimos concluir que de la competencia identificada ninguno ofrece un sistema integral que incluya aplicación mobile, terminal autoservicio, web de pedidos, terminal de comandas y *backoffice* web. Por lo que el sistema desarrollado es más completo y eso implica que todos los pedidos serán procesados por un solo programa, lo que deriva en gráficas e información más acertada.

4.3.1.5. Reuniones de extracción con usuarios

Metodología y motivación

Se realizaron tres reuniones con un mismo grupo de usuarios que cumplen con el perfil objetivo del proyecto. Estás se hicieron de manera presencial con el fin de analizar las funcionalidades desarrolladas en cada versión y extraer posibles cambios o mejoras. Se hizo una primera reunión en una etapa temprana del proyecto basándonos en los prototipos, luego las otras dos fueron al finalizar el *release* uno y dos.

El fin de cada reunión era poder obtener distintas opiniones sobre la solución hasta el momento, observar distintas experiencias y que los participantes nos transmitan posibles sugerencias.

Procedimiento

Antes de cada reunión el equipo se preparaba planteando los objetivos de cada una teniendo en cuenta que información se buscaba obtener. Para esto se redactan preguntas como por ejemplo, qué solución se imagina respecto a cierta funcionalidad, que cambios le haría a la navegación para que le quede más cómodo, entre otras dependiendo la instancia.

Previamente, también se definía un integrante del equipo para que lleve adelante el proceso utilizando una guía para no perder el rumbo y respetar algunas pautas. Los demás integrantes cumplían un rol de apoyo ante cualquier inconveniente o duda de los usuarios y tomaban apuntes de sus aportes.

Al finalizar cada una de estas sesiones el equipo se reunía para analizar todo lo obtenido, se buscaba sacar conclusiones basadas en los objetivos antes definidos y plantear las ideas en nuevas funcionalidades.

Resultados

Aquí se pueden ver algunas de las sugerencias principales que se obtuvieron luego de cada reunión con los usuarios.

Primera reunión con prototipos:

- Imágenes muy grandes.
- Contraste de letras del precio y nombre de producto en las *cards* de la pantalla principal.
- Agregar un menú de opciones en la parte inferior de la pantalla.

Segunda reunión después de primer *release*:

- Agregar promociones destacadas.
- Categorías de comidas arriba de los productos para hacer un filtrado más rápido.
- Poder hacer pedidos desde la caja en una opción en el *backoffice.*

Tercera reunión después de segundo *release*:

- Que no sea necesario salir de la pantalla principal para agregar un nuevo producto, para esto se adapta un bottom *modal sheet* desde donde se pueden agregar nuevos productos al carrito.
- En pantalla de comandas poder volver del estado “Preparación” o “Finalizado” a “Pendiente”.
- En pantalla de comandas visualizar hace cuánto tiempo llegó la comanda.
- En *dashboard* agregar promociones más utilizadas y ventas por plataforma.
- Que el usuario pueda volver a repetir pedidos que hizo en algún otro momento.

4.3.2. Especificación

Una vez obtenidos los requerimientos, estos son especificados en forma de *user stories* cómo se hace en el *framework Scrum*. En ellas se describe de manera informal una función escrita desde la perspectiva del usuario final. También en cada una se incluye su estimación planeada, la prioridad, otros archivos de interés y criterios de aceptación.

Se encuentran varias ventajas en su uso, algunas de ellas son: permitir mantener el foco en resolver problemas para usuarios reales, permiten el trabajo colaborativo, impulsar soluciones creativas, motivación al realizar pequeños desafíos y tener pequeñas “victorias”.

A modo de ejemplo se incluye una historia de usuario en la herramienta *Jira*:

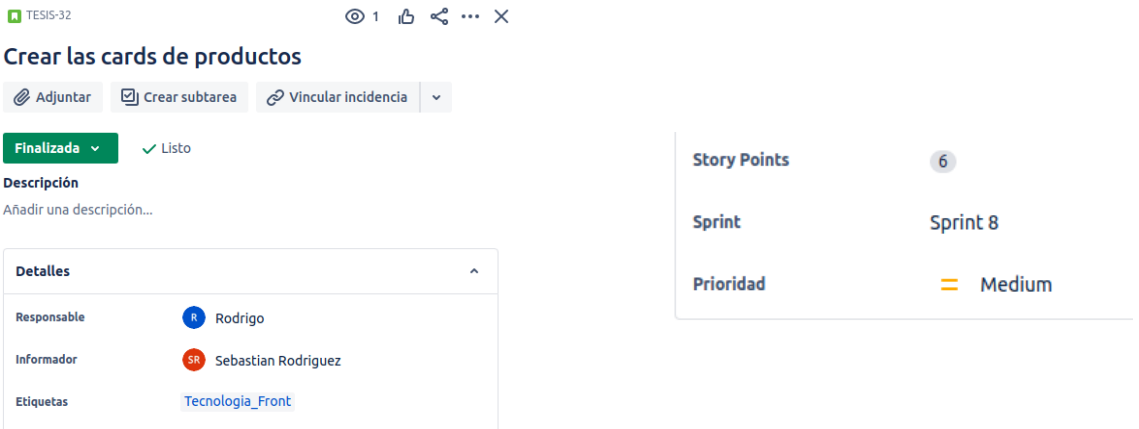


Figura 51: Especificación de requerimientos

4.3.3. Validación

En la siguiente sección se presentan distintas actividades que se realizaron para validar los requerimientos relevados en el proceso de ingeniería de requerimientos.

4.3.3.1. Prototipos

Para obtener *feedback* rápido sobre decisiones tanto de UX como de UI se utilizaron distintos prototipos de acuerdo a las necesidades del requerimiento en validación.

Para esto utilizamos una herramienta llamada Figma para lograr un diseño lo más fidedigno posible, pero también en un inicio se realizaron bosquejos más sencillos. De esta manera es que logramos mostrar posibles cambios o mejoras a usuarios y al cliente e ir puliendo detalles hasta llegar a lo requerido.

- Prototipo móvil

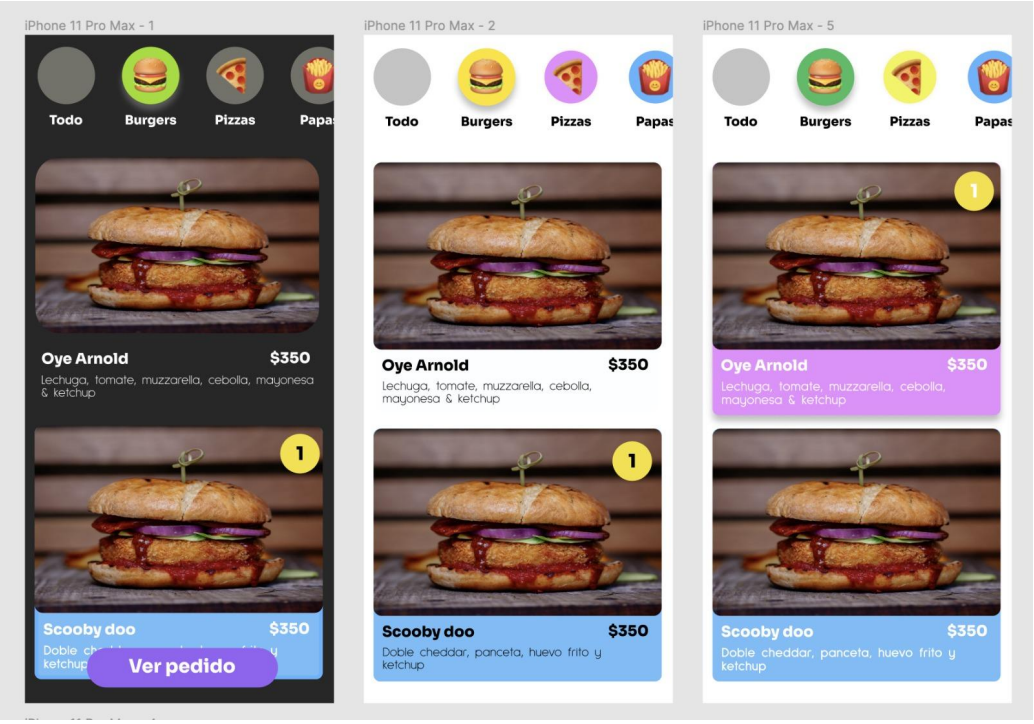


Figura 52: Prototipo móvil

- Prototipo para cocina

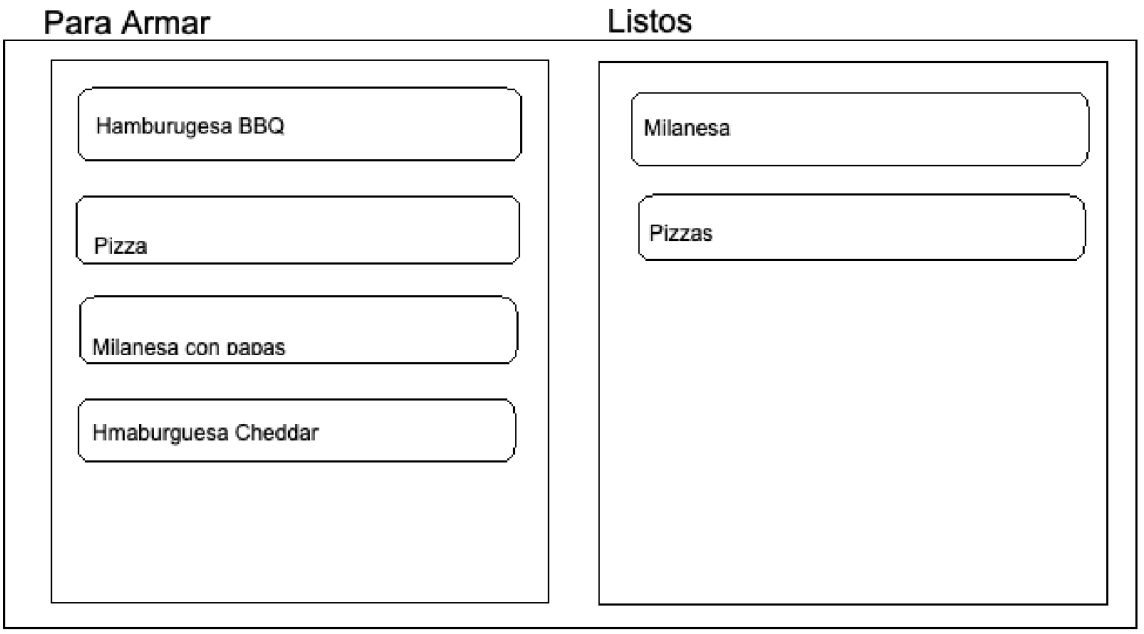


Figura 53: Prototipo cocina

- Prototipo backoffice web

Figura 54: Prototipo backoffice

4.3.3.2. Validación con usuarios reales

Como parte del proceso de validación, el equipo realizó pruebas de campo con grupos formados por los identificados como usuarios finales con el fin de obtener retroalimentación sobre las decisiones tomadas, y si es necesario, se realizan correcciones o mejoras.

Estas pruebas se realizaron en prototipos para hacer validaciones tempranas de requerimientos y sobre la plataforma para validar requerimientos ya implementados.

Al igual que las pruebas con usuarios en la etapa de extracción, estas se llevaron a cabo con el mismo grupo de usuarios en varias instancias.

Antes de cada reunión se plantean los objetivos a validar, el equipo se coordina para llevar adelante las pruebas de manera ordenada y tomar notas de las observaciones.

En algún caso fue necesario plantear a los usuarios más de una opción de un requerimiento cuando surgían dudas y más de una solución a considerar. Estas se mostraban en forma de prototipo para que ellos emitan su opinión y nosotros luego poder llegar a una decisión acorde a la satisfacción del usuario.

Se tuvo en cuenta los comentarios realizados por los usuarios y al poder realizar las pruebas de manera presencial se tomaba apuntes del lenguaje corporal expresado.

En las últimas instancias los objetivos estaban enfocados en validar detalles y evaluar la usabilidad de algunos requerimientos. Por lo que la dinámica de las pruebas consistió en definir que cada usuario utilice los distintos flujos de la plataforma intentando de simular en completitud su funcionamiento en la realidad.

Complementando lo anterior, en cada una de las pruebas se midió las métricas de tiempo promedio y cantidad de clicks para un flujo completo, con el fin de evaluar efectivamente sí se encontraron mejoras en los diferentes *releases*. Un análisis más detallado puede verse en la sección 8.5. Métricas.

4.3.3.3. Validaciones con el cliente

Para concluir con el proceso de ingeniería de requerimientos el equipo consideró la necesidad de tener la aprobación del cliente sobre los requerimientos especificados.

Para esto se realizaron reuniones en conjunto con el cliente donde el equipo presentaba los requerimientos funcionales y no funcionales para discutir y terminar de concluir que lo obtenido correspondía con sus expectativas.

De esta manera se podía comprobar que el cliente se encontraba satisfecho con la especificación realizada.

4.4. Actores

Se presentan los actores identificados que interactúan en los distintos procesos del sistema, estos son:

Administradores del bar: Son las encargadas de gestionar el negocio y mantener toda la información de la aplicación.

Cocinero: Funcionario encargado de seleccionar las comandas para cocinar y actualizar su estado.

Cajero: Funcionario que se encuentra en la caja recibiendo nuevos pedidos por parte de los clientes.

Cliente: Es quien va a asistir al bar y utilizar la solución para hacer pedidos.

Mercado Pago: Gestor de pagos que utiliza el sistema para cobrar los pedidos a los clientes.

4.5. Requerimientos Funcionales

Luego de tener identificados los procesos del negocio y sus actores es posible relacionarlos con los requerimientos funcionales recabados. A continuación, se listan y se detalla que actores se incluyen en cada uno.

Aplicación Backoffice		
Id	**Requerimiento**	**Actor**
RF1	Gestión de usuarios	Administradores del bar
RF2	Gestión de productos	Administradores del bar
RF3	Gestión de categorías	Administradores del bar
RF4	Gestión de promociones	Administradores del bar
RF5	Gestión de cupones	Administradores del bar
RF7	Gestión de comandas	Administradores del bar
RF8	Gestión de stock	Administradores del bar
RF9	Gestión comandas en efectivo	Administradores del bar / Cajero
RF10	Mercadopago - Actualizar status pedido	MercadoPago
RF11	Pantalla dashboard	Administradores del bar
RF12	Pantalla caja	Administradores del bar / Cajero
RF13	Envío de notificaciones	MercadoPago / Administradores del bar
RF14	Enviar email al recuperar contraseña	Cliente

Tabla 1: Requerimientos Funcionales de aplicación backoffice.

Aplicación Móvil		
Id	**Requerimiento**	**Actor**
RF15	Login en aplicación	Cliente
RF16	Registro de usuario	Cliente

RF17	Recuperar contraseña	Cliente
RF18	Menú	Cliente
RF19	Agregar producto	Cliente
RF20	Carrito	Cliente
RF21	Selección de bipper	Cliente
RF22	Perfil de usuario	Cliente
RF23	Editar perfil de usuario	Cliente
RF24	Histórico de pedidos	Cliente
RF25	Repetir pedidos	Cliente
RF26	Pedidos sin pagar	Cliente
RF27	Comandas	Cocinero / Cliente
RF28	Pantalla de promociones	Administradores del bar / Cliente
RF29	Pago con Mercadopago	Cliente / MercadoPago
RF30	Pago con efectivo	Cliente / Cajero
RF31	Aplicar promociones	Administradores del bar / Cliente
RF32	Ingreso cupones	Administradores del bar / Cliente
RF33	Flutter web	Cajero
RF34	Filtro categorías	Cliente
RF35	Push notifications	Administradores del bar / Cajero / Cocinero / Cliente

Tabla 2: Requerimientos Funcionales de aplicación móvil.

4.6. Requerimientos No Funcionales

Como se menciona en secciones anteriores, el cliente no cuenta con gran entendimiento tecnológico por lo que no han impuesto ningún requerimiento no funcional. Igualmente trabajando en conjunto con ellos y de lo recabado en la etapa de especificación se pudo identificar los siguientes.

Usabilidad	
Id	**Requerimiento**
RNF1	Tanto app mobile como backoffice deben ser responsive y verse bien en todos los tamaños de dispositivos.
RNF2	Debe ser usable para los clientes, cumpliendo con las heurísticas de Nielsen.

Tabla 3: Requerimientos No Funcionales usabilidad.

Modificabilidad	
Id	**Requerimiento**
RNF3	Debe ser configurable.
RNF4	Fácil modificación en el desarrollo, al ser ciclo iterativo e incremental se debe poder incorporar nuevos requerimientos que vayan surgiendo o cambios. Cuanto más mantenible menos impacto.
RNF5	Seguir estándares y convenciones de codificación.

Tabla 4: Requerimientos No Funcionales modificabilidad.

Portabilidad	
Id	**Requerimiento**
RNF6	Los sistemas deben correr sin problemas tanto en iOS como Android y en distintas terminales.

RNF7	El back office debe ser soportado en Chrome, Safari y Firefox.

Tabla 5: Requerimientos No Funcionales, portabilidad.

Seguridad	
Id	**Requerimiento**
RNF8	Los usuarios para acceder deben estar autenticados.
RNF9	Los usuarios deberán tener roles para poder acceder a las distintas partes de la aplicación.
RNF10	La transferencia de datos entre servidores debe ser cifrada.

Tabla 6: Requerimientos No Funcionales, seguridad.

Disponibilidad	
Id	**Requerimiento**
RNF11	El sistema debe contar con un uptime de 90% o mayor.
RNF12	El equipo debe responder en no más de 24 horas ante fallas o errores.

Tabla 7: Requerimientos No Funcionales disponibilidad.

Performance	
Id	**Requerimiento**
RNF13	Las peticiones deben ejecutarse en menos de 3 segundos.
RNF14	Las comandas deben llegar a la cocina en tiempo real.

Tabla 8: Requerimientos No Funcionales, performance.

5. Arquitectura

5.1. Descripción General

En este capítulo se especificarán las diferentes decisiones arquitectónicas tomadas sobre la composición de la arquitectura y el diseño de Random App. En esta sección, se encontrará la justificación de la implementación de cada uno de los requerimientos funcionales, no funcionales y restricciones del sistema.

La solución está conformada por los siguientes componentes:

1. Aplicación móvil desarrollada para sistemas operativos *iOS* y *Android.*
2. Terminal de autoservicio desarrollada para sistemas operativos Android.
3. Aplicación para tablet (centro de comandas) para sistemas operativos Android.
4. Aplicación Web de pedidos.
5. *Backoffice* para la administración del sistema
6. *Backend* el cual a través de una *API REST* expone los servicios a utilizar en las aplicaciones mencionadas en los puntos anteriores.

5.2. Diagrama de Contexto

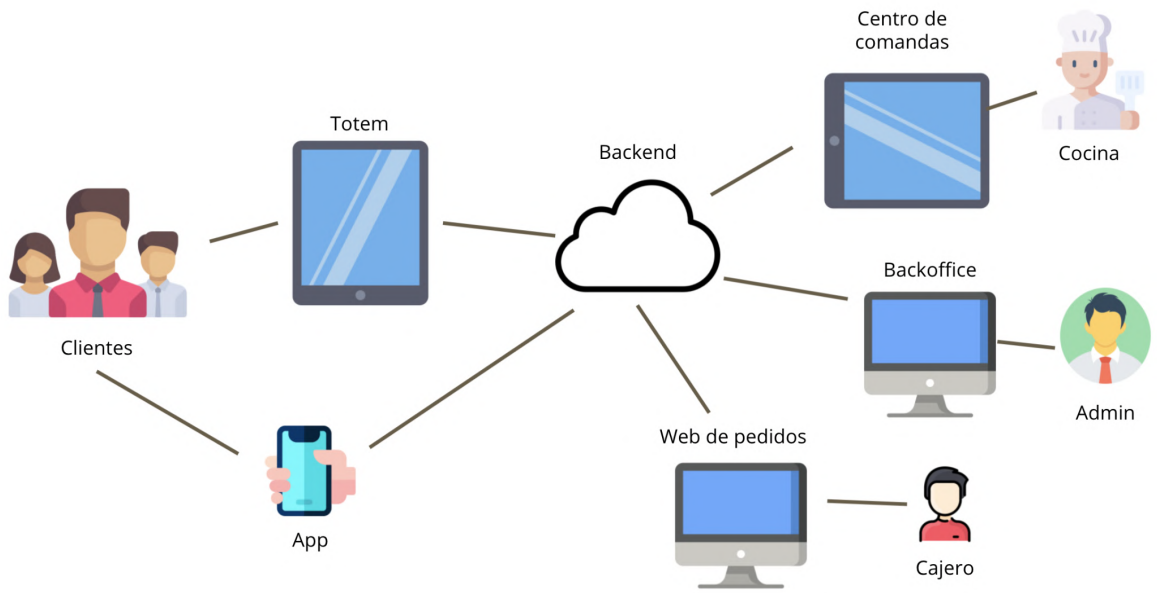


Figura 55: Diagrama de contexto

5.3. Descripción de la arquitectura

En esta sección se describirán las diferentes decisiones que el equipo ha tomado para diseñar la arquitectura. La principal decisión fue la de realizar una arquitectura basada en un único *backend* centralizado. Este *backend* es consumido por los distintos componentes del sistema, desde las aplicaciones web, la terminal de autoservicio como también la web para administradores del sistema. Por otra parte, el *frontend* y el *backend* están ubicados en distintos servidores ya que de esta forma se evita que los cambios en *frontend* no repercutan directamente sobre el *backend* y sus funcionalidades.

Una vez tomadas estas decisiones, lo siguiente que el equipo debió decidir fue que tipo de estrategia de arquitectura y de despliegue se utilizará dadas las características del proyecto. Finalmente se optó por una arquitectura monolítica, a continuación, se detallan las razones.

Una arquitectura monolítica [51] consiste en tener todo el *backend* en un solo servidor. A partir de esto, se analiza la solución teniendo en cuenta diferentes aspectos tales como el costo de desarrollo, el costo de mantenimiento y la performance con respecto a una solución basada en microservicios.

Sabiendo cuales son los beneficios de los microservicios sobre las soluciones monolíticas, tales como una mejor modificabilidad en los sistemas y su mejor escalabilidad ya que no es necesario replicar el *backend* en su totalidad, de igual forma se optó por una solución monolítica debido a las siguientes consideraciones.

Una arquitectura monolítica hace simple y veloz el desarrollo de un sistema dado que se contiene el código fuente en un único proyecto. En contraposición a la implementación de una arquitectura basada en microservicios que implica la creación de diversas interfaces y distintos canales de comunicación entre cada uno de ellos.

Por otro lado, el costo de mantenimiento de una arquitectura monolítica, es menor a la de microservicios ya que se encuentra todo centralizado a diferencia de los microservicios que se encuentran distribuidos.

Por lo mismo, es que una arquitectura monolítica es más performante ya que la comunicación y búsqueda de información es más rápida favoreciendo así la usabilidad del sistema.

5.4. Diagrama de componentes y conectores

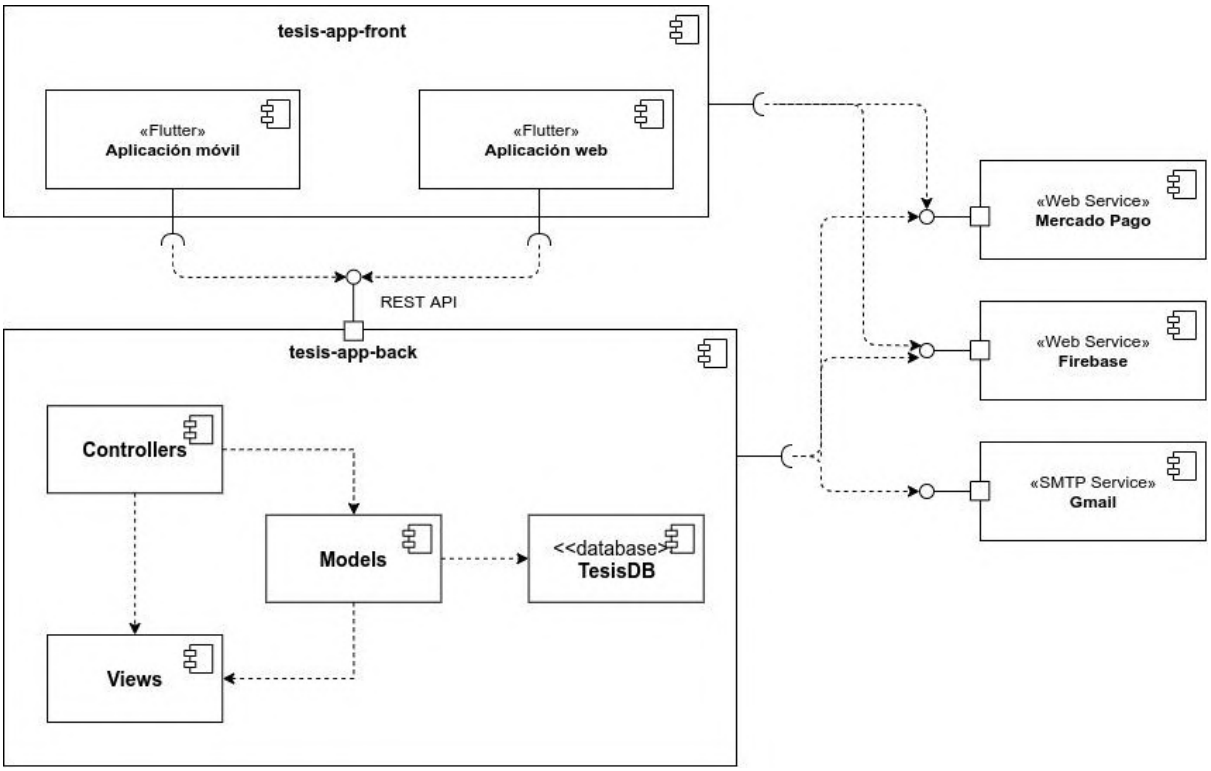


Figura 56: Diagrama de componentes y conectores

5.4.1. Catálogo de Componentes

Aplicación Móvil

Esta es la aplicación para los usuarios que se encuentren en el bar. La misma fue desarrollada en *Flutter* y cumple la funcionalidad de que los usuarios puedan realizar los pedidos sentados desde la mesa.

Terminal de autoservicio

Corresponde a la aplicación la cual se puede correr en tótems ubicados en lugares estratégicos del bar o también se puede instalar en tablets para situar en cada una

de las mesas. Esta aplicación fue desarrollada en *Flutter* y les permite a los usuarios del bar realizar el pedido que deseen.

Con el fin de que cada cliente pueda ser notificado cuando su pedido esté pronto, él mismo recogerá un *beeper* el cual estará situado a un costado de la terminal y lo vincula a su orden ingresando el número del mismo en la terminal de autoservicio.

Aplicación para tablet (centro de comandas)

Esta aplicación se diseñó con el fin de que los cocineros tengan una pantalla en la cocina en la cual pudieran tener en tiempo real todos los nuevos pedidos y al mismo tiempo ir manejando los que se están preparando. De igual forma que las anteriores, esta aplicación fue desarrollada en *Flutter* y permite a los cocineros trabajar de una forma más ordenada.

Backoffice para la administración del sistema

Corresponde a la aplicación para administradores del sistema. Esta aplicación fue desarrollada en *Rails* y desde la misma se tiene control sobre todos los distintos elementos que figuran tanto en la aplicación móvil como en la terminal de autoservicio. A su vez tiene funcionalidades extras tales como el manejo de stock del bar.

Aplicación web de pedidos

Con el fin de que el cajero pueda utilizar el mismo sistema que la terminal autoservicio y que la aplicación móvil, se desarrolló en *Flutter* una aplicación web con el mismo flujo que la terminal y la aplicación con el fin de poder tener todos los pedidos unificados en un solo sistema.

Firebase

Es el servicio utilizado para el envío de *notificaciones push* a la aplicación móvil y también como servicio para el *hosting* de la aplicación para la versión que se encuentra dentro del sistema de administradores.

Mercado Pago

Servicio utilizado para el manejo de todos los pagos que no son en efectivo dentro de la plataforma.

Backend Random House App

Backend de la solución. Es el encargado de procesar todas las solicitudes y también de comunicar a todos los distintos componentes del sistema. Fue desarrollado en *Ruby on Rails* y deployado en *Heroku.*

Base de datos PostgreSQL

Es la base de datos utilizada para guardar toda la información del sistema. Se decidió optar por esta tecnología ya que es gratuita e ilimitada, además de permitir una gran escalabilidad. Esta base de datos tiene más de 20 años de desarrollo lo cual ofrece estabilidad, confiabilidad y robustez.

Se decidió por una base de datos relacional ya que, por la naturaleza del proyecto, los datos estarían más ordenados y realizar consultas sería más sencillo.

Se puede ver el diagrama de la base de datos en el Anexo 12.6. Diagrama de base de datos

Gmail

Servidor SMTP para el envío de emails. En el sistema se usa para la recuperación de contraseña.

5.5. Diagrama de deploy

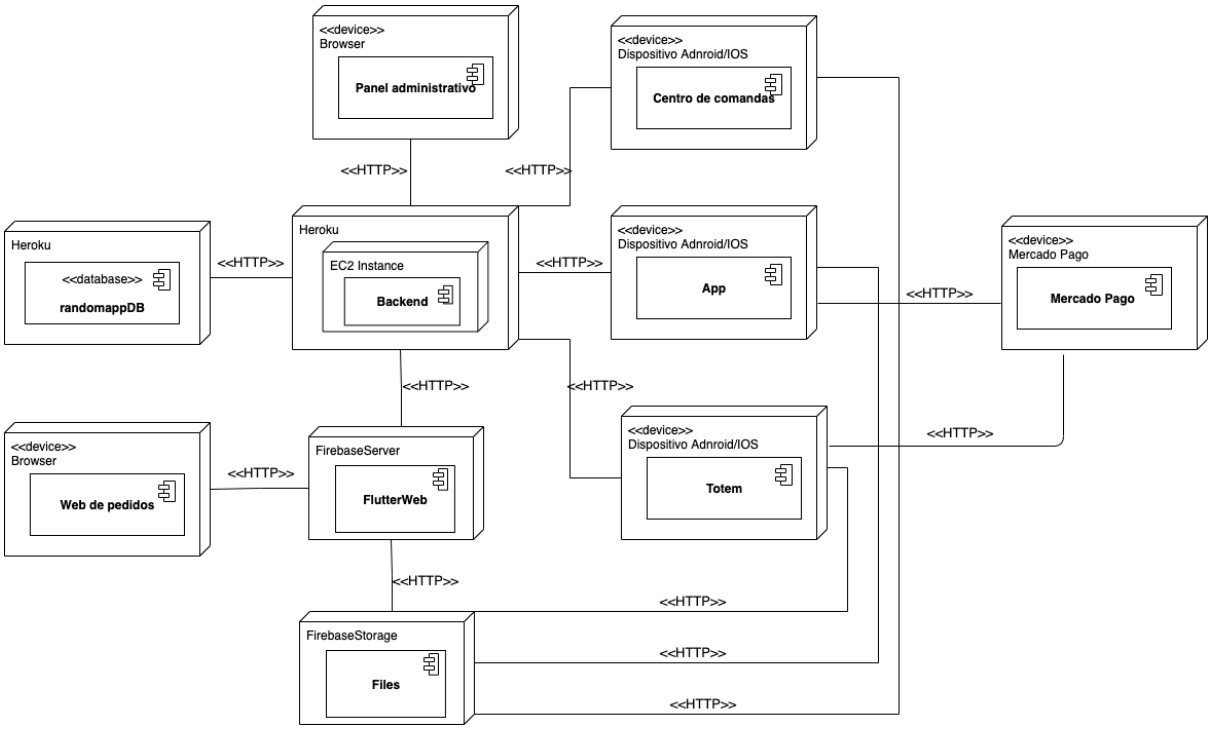


Figura 57: Diagrama de deploy

5.6. Elección de tecnologías

En esta sección se describen cuáles fueron las distintas tecnologías utilizadas en los distintos componentes, y se explicará por qué se optó por la misma. Para estas elecciones los integrantes investigaron las distintas tecnologías que se encuentran en el mercado y evaluaron cuales eran más acorde a la solución del sistema. Hay que tener en cuenta que el cliente no es experto en el tema por lo cual estas decisiones fueron tomadas en su totalidad por decisión propia de los integrantes del equipo.

5.6.1 Criterios de selección de tecnologías

Tanto para la parte de *backend* y *frontend*, se definieron varios criterios generales a tener en cuenta para seleccionar las tecnologías de desarrollo adecuadas. Estos son:

Simplicidad de uso y rápido aprendizaje

El proyecto debe ser desafiante desde todos los aspectos. Igualmente se tiene en cuenta que es un proyecto de grado y los tiempos de desarrollo no son los más largos. También el equipo está compuesto por tres integrantes, los tres van a tener que utilizar todas las tecnologías. Por esto es de suma importancia que en un corto tiempo el equipo sea capaz de aprender a utilizar las tecnologías y así poder comenzar a desarrollar las primeras funcionalidades rápidamente.

Creación rápida de un mínimo producto viable

Para las características del proyecto, se considera importante poder contar con un mínimo producto viable lo más rápido posible. El equipo requiere de esto para poder validar la información obtenida de los usuarios con el menor esfuerzo posible. Esto va de la mano con poder crear nuevas versiones rápidamente.

Experiencia

Los integrantes del equipo cuentan con un buen conocimiento de algunas tecnologías por razones laborales. Por lo que se decidió en lo posible no utilizar estas para ampliar el conocimiento y desafío del proyecto.

Experiencia de usuario

Las tecnologías deben permitir crear pantallas intuitivas para el usuario. Su arquitectura debe permitir una completa personalización resultando en un renderizado rápido y con diseños expresivos y flexibles. La aplicación móvil es soportada en iOS y Android, por lo que se valora que se permita compartir componentes entre ambas plataformas como scrolling, navegación, iconos y fuentes.

Comunidad

Es de suma importancia que las tecnologías cuenten con una gran comunidad. Ante cualquier problema se puede contar con respuestas rápidas. Muchos de los posibles *bugs* se resuelven rápidamente. El producto necesita tener capacidad de integración

con terceros, por lo que cuanto más grande sea la comunidad más capacidad de integración tiene.

Portabilidad

Es necesario que la aplicación móvil realizada sea soportada en Android e iOS. Por lo que cuanto más se puedan reutilizar los componentes de código en ambos, el desarrollo va a ser más rápido, sencillo y mantenible.

Costo

Todas las tecnologías deben ser gratuitas.

5.6.2. Análisis de tecnología móvil a utilizar

A continuación, se detallan las razones por las cuales se seleccionó la tecnología para el desarrollo de la aplicación móvil.

El equipo en un principio consideró tres opciones posibles para el desarrollo de la aplicación móvil. Estas opciones eran, por un lado, un desarrollo nativo, por otro lado, un híbrido, y por último una aplicación web progresiva.

Para decidir cuál de estas opciones se implementaría, el equipo hizo especial énfasis en cómo es la solución del sistema. La misma cuenta con una aplicación móvil para que los clientes del bar la utilicen en sus celulares, otra aplicación la cual será utilizada en la terminal de autoservicio y por último una aplicación la cual será instalada en las tablets de la cocina para el manejo de las comandas.

Dada esta situación, el equipo rápidamente descartó la opción de hacer un desarrollo nativo. Esto implicaría hacer un desarrollo específico para cada sistema operativo (iOS y Android) de forma independiente, y dados los tiempos del proyecto fue descartada esta opción.

Por otro lado, se manejó la opción de desarrollar una aplicación web progresiva. La misma constaba de una aplicación web desarrollada en HTML, CSS y JavaScript la cual tiene como ventaja el hecho de que se ejecuta a través de un navegador web sin la necesidad de tener que ser instalada. Al mismo tiempo, al ser multiplataforma, solo se tendría que desarrollar un único código lo cual es una ventaja con respecto a

la opción nativa. Si bien esta era una opción muy tentadora, la debilidad más grande es la performance lo que impacta directamente en la usabilidad de la aplicación, que es el atributo de calidad en el cual equipo no estaba dispuesto a ceder ya que es clave que el usuario tenga una excelente experiencia a la hora de utilizar la aplicación.

Dada las dos situaciones planteadas, se analizó la opción de una aplicación híbrida, la cual permite crear aplicaciones móviles nativas tanto para Android como iOS con un único código base. Lo anterior mencionado es uno de los beneficios más grandes ya que al tener un único código para las distintas plataformas se ve favorecida la mantenibilidad y al mismo tiempo incrementa la productividad del equipo.

Si bien este es un beneficio que la aplicación web progresiva también tiene, la diferencia radica en que al desarrollar una aplicación híbrida la performance de la aplicación no se ve afectada y al mismo tiempo, le brinda al usuario una experiencia más nativa dentro de la aplicación lo cual hace que la usabilidad se vea favorecida. Es por esto que el equipo terminó optando por el desarrollo híbrido ya que es la opción que más se adapta a las necesidades de nuestro proyecto, beneficiando distintos aspectos del mismo.

Una vez decidido el enfoque a utilizar, se evaluó en qué tecnología se iba a desarrollar.

Por un lado, tenemos *Flutter*, una tecnología desarrollada por Google y por otro lado tenemos React Native, tecnología desarrollada por Facebook.

A continuación, se deja evidencia de una tabla comparativa entre las dos tecnologías.

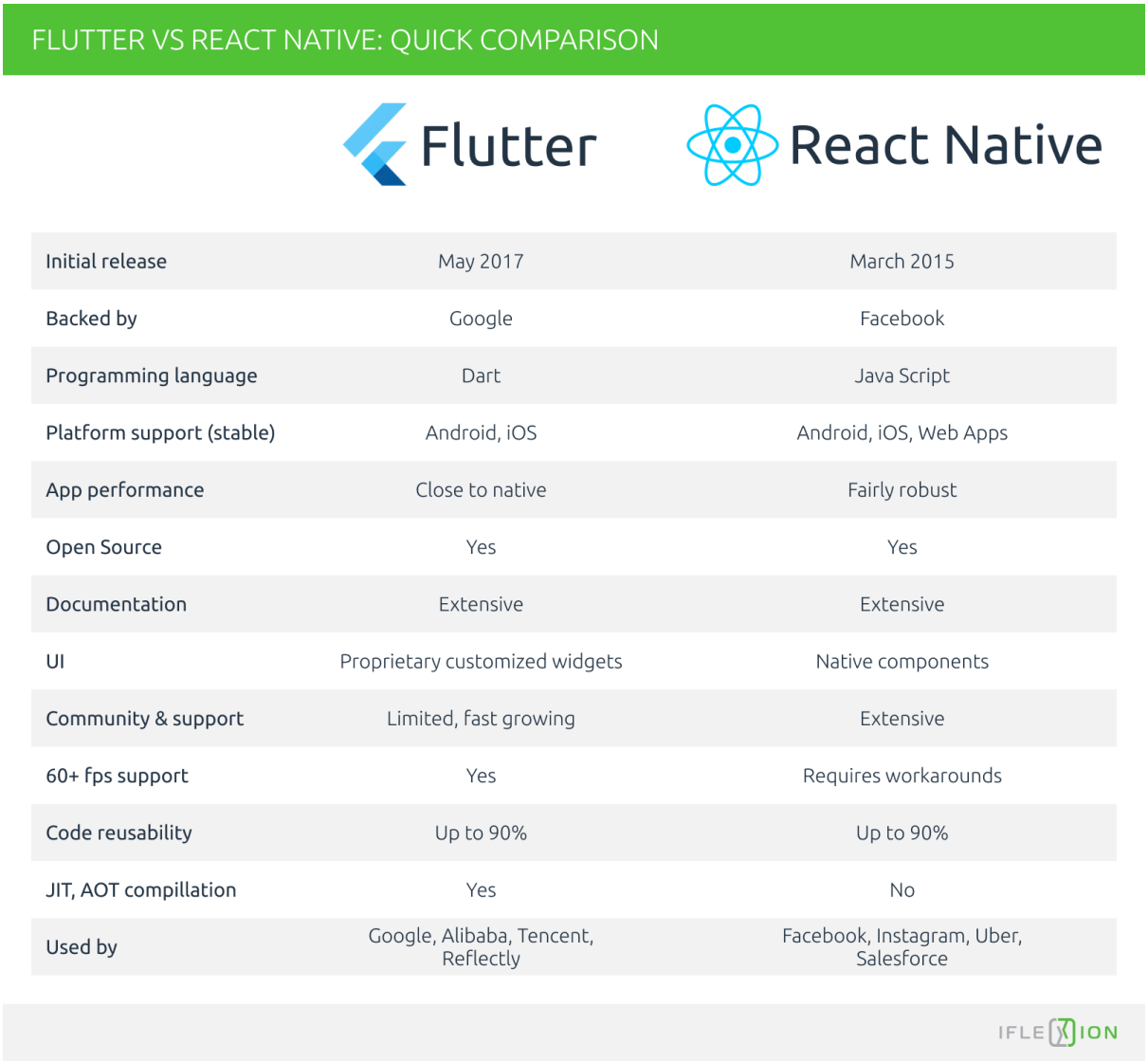


Figura 58: Comparativa Flutter vs React Native [52]

Una vez analizado cada uno de los puntos comparativos, y aplicándolos en nuestro proyecto, se llegó a la conclusión de que *Flutter* es el que mejor se adapta. Por otro lado, otra ventaja que presenta *Flutter* es que uno de los integrantes ya poseía experiencia previa, mientras que en React Native ninguno había utilizado.

Terminal de autoservicio y aplicación móvil

La terminal autoservicio tiene ciertas diferencias con la aplicación móvil pero el flujo principal de realizar un pedido es el mismo en ambas. Es por esto que se diseñaron estas dos aplicaciones en conjunto y mediante un manejo de roles con distintos permisos se accede al “modo terminal autoservicio”.

Centro de comandas

Luego de realizar un análisis, ir al bar a ver el lugar donde sería colocada la terminal de comandas y hablar con las personas que utilizarían este sistema, se llegó a la conclusión de que la mejor forma de implementar esta aplicación era en una tablet y no en una PC. Es por esto, que al igual que la aplicación móvil y que la terminal autoservicio, el centro de comandas está desarrollado con *Flutter*, para así poder correr de forma nativa en cualquier tablet Android o iOS.

5.6.3. Análisis de tecnología backend a utilizar

Se presentan las opciones que el equipo tuvo en consideración antes de comenzar con el desarrollo del *backend*, sus comparaciones y razones de la selección.

Al ser libre la selección surgieron varias opciones, entre ellas las más destacadas fueron Django, *Ruby on Rails* y Node.js.

Node.js

Es una plataforma de código abierto para ejecutar código JavaScript del lado del servidor, que además fue creada con el runtime de JavaScript. Es un entorno controlado por eventos permitiendo establecer y gestionar múltiples conexiones al mismo tiempo, esto hace que no sea necesario preocuparse por bloqueos. Gracias a estas características Node.js tiende a ser más eficiente y liviano. Algunos de los aspectos que tiene a favor son:

- Se puede utilizar el mismo lenguaje del lado del cliente y el servidor, si en el desarrollo de *frontend* Javascript es muy usado este es un punto importante.
- Es bueno para microservicios por ser escalable y liviano.
- Flexibilidad de desarrollo.
- Entradas y salidas sin bloqueo permitiendo procesar numerosas cantidades de solicitudes al mismo tiempo.
- Bueno para aplicaciones de tiempo real como apps de mensajería, juegos online, plataformas de colaboración.
- Cuenta con una gran comunidad.

Ruby on Rails

Es un *framework* de aplicaciones web de código abierto escrito en el lenguaje *Ruby*, siguiendo el patrón de arquitectura Modelo Vista Controlador (MVC). Intenta combinar la simplicidad con la posibilidad de desarrollar aplicaciones escribiendo menos código que con otros *frameworks* y con un mínimo de configuración. Estos son los aspectos a favor más importantes:

- Es un *framework* fuertemente estructurado. *Rails* cuenta con muchas reglas para organizar el código. Usa el patrón MVC, obligando a crear view, controller y ruta para la creación de una petición. Esto resulta en un código muy bien organizado, fácil de leer, de mantener y editar.
- Cuenta con todo lo necesario para crear una aplicación web basada en base de datos pronta para usar. Esto incluye un ORM (mapeo relacional de objetos), desarrollo de *API* RESTful, *scaffolding*, administración del entorno, procesamiento en segundo plano, testing y comunicación en tiempo real mediante *websockets*. Dado que las características básicas de las aplicaciones CRUD se pueden generar fácilmente sin escribir mucho código, *Rails* es muy apto para la creación rápida de prototipos y el desarrollo de *MVPs*.
- Cuenta con muchos recursos de código abierto (gemas) disponibles en caso de que sea necesario el desarrollo fuera de lo convencional.
- Se adapta muy bien a metodologías ágiles de desarrollo de software.
- Cuenta con una gran comunidad.

Django

Es un *framework* de desarrollo de aplicaciones web de código abierto escrito en Python, respeta el patrón de arquitectura modelo-vista-controlador. Django se inventó para cumplir con plazos de desarrollo apresurados, y al mismo tiempo, satisfacer las altas expectativas y requisitos de desarrollos web complejos. Estos son algunas de sus fortalezas:

- Es un software maduro, cada parte de su código fue examinado y probado por una gran cantidad de desarrolladores.
- Cuenta con una gran cantidad de paquetes de terceros integrables.
- Al igual que *Ruby on Rails* utiliza el patrón MVC.

- Énfasis en el reúso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio “No te repitas” (*DRY*, del inglés *Don't Repeat Yourself*).
- Cuenta con una gran comunidad.

Decisión

Al tomar la decisión final se tuvo en cuenta los criterios de selección de tecnologías. Para el *backend* se destaca la necesidad de desarrollo veloz para liberar un primer *MVP* y desarrollar nuevas versiones. También se considera la experiencia previa del equipo con las tecnologías y los objetivos planteados con los atributos de calidad.

Django cuenta con características que hacen foco en el criterio de velocidad de desarrollo. De igual manera, el equipo no cuenta con ningún tipo de experiencia previa en el uso del *framework*, por lo que el equipo creyó que lo mejor sería descartarlo.

Por otro lado, se encuentra Node.js, con muy buenas características que quizás alienten su uso, es un lenguaje muy popular con muchas posibilidades de crecimiento. Su gran desventaja, en el contexto del proyecto, es el tiempo de desarrollo para alcanzar un mínimo producto viable rápidamente e incorporar nuevas versiones.

Luego se encuentra *Ruby on Rails*, un *framework* muy bien estructurado. Su mayor desventaja puede decirse que es su performance en comparación con otros, pero esto no quiere decir que no sea suficiente para la solución. Al igual que Django, *Ruby on Rails* es muy bueno en cuanto a productividad de desarrollo.

Por razones de estudio, el equipo cuenta con experiencias previas en las últimas dos tecnologías mencionadas, esto es importante porque se tiene nociones de como comenzar y de igual manera sigue siendo desafiante.

Se consideró utilizar el *framework* Express.js para programar con el lenguaje Node.js, esto podría solucionar el problema de productividad que tiene Node.js. Pero luego de investigar un poco más se encontraron algunas desventajas. Se presenta una tabla de comparación.

	Ruby on Rails	Express.js
Velocidad de programación	Muy útil cuando se quiere crear aplicaciones rápidamente y probar las ideas.	Requiere más tiempo en empezar y probar la app.
Performance	No puede soportar tantas peticiones a la vez como Express.js.	Es conocido por la velocidad gracias a la combinación de la programación orientada a eventos y la asincronía.
Popularidad	Usado en 381,367 sitios web. Cuenta con una gran cantidad de desarrolladores seniors en su comunidad.	Usado en 174,112 sitios web. El nivel de desarrolladores en la comunidad es muy variado.
Documentación	Una tecnología madura. Por lo tanto, tiene una comunidad más grande que conduce a una documentación más amplia.	A veces los módulos son de calidad pobre y la documentación carece de especificaciones.
Curva de aprendizaje	Fácil de empezar.	Una buena opción para desarrolladores con conocimiento en Node.js.
Comunidad y soporte	Apoyado por una comunidad de desarrolladores activa y solidaria.	Es una tecnología más joven que no puede compararse en términos de soporte contra RoR.

Tabla 9: Comparación de tecnologías.

En modo de conclusión, se tuvieron en cuenta todas las ventajas y desventajas de optar por una u otra y se llegó a que la mejor opción es usar *Ruby on Rails* para la programación del *backend*. Teniendo en cuenta que para el producto los aspectos

de escalabilidad y performance son cubiertos por *Rails*, la decisión terminó definiéndose por la velocidad de inicio y productividad al correr de todo el proyecto, siendo la más indicada *Ruby on Rails*.

Esta tecnología está construida sobre el patrón *Model-View-Controller (MVC)*, el cual favorece uno de los atributos de calidad del proyecto tal como la modificabilidad. Al mismo tiempo, es muy sencillo de integrar con librerías de terceros. Prácticamente existe cualquier tipo de librería la cual esté en formato de *gema* la cual puede ser instalada en la aplicación mediante la herramienta *Ruby Gems*.

Por otro lado, *Ruby on Rails* es una de las tecnologías con mayor popularidad en el sitio *GitHub*. Cuenta con una comunidad muy amplia de desarrolladores los cuales están constantemente mejorando código y ayudando a la comunidad a resolver sus problemas. Esta es una gran ventaja ya que simplemente uno de los integrantes del equipo maneja experiencia previa en esta tecnología, por lo cual para los otros dos, el conseguir documentación y ejemplos de código no será un problema, favoreciendo así la curva de aprendizaje.

5.6.4. Aplicación Web

Debido a la utilización de *Ruby on Rails* para la implementación del *backend*, y como se mencionó anteriormente, esta tecnología está construida sobre el patrón *Model-View-Controller (MVC)*, se aprovechó esta ventaja para utilizar las vistas que *rails* genera para utilizarlas en la aplicación web.

La principal funcionalidad de la aplicación web es funcionar como un sistema administrativo para la gestión de los diferentes modelos que se manejan en el sistema. Es por esto que no fue necesario la utilización de otras tecnologías las cuales nos puedan brindar funcionalidades extras ya que con las vistas que *rails* genera utilizando HTML y CSS fue más que suficiente para generar un buen panel administrativo.

Además de la aplicación web mencionada se desarrolló otra para la toma de comandas en desde la caja, la misma fue desarrollada en *Flutter* ya que al igual que la terminal de autoservicio y la aplicación móvil el flujo de realizar un pedido es

similar. Esto reforzó utilizar *Flutter* como tecnología a utilizar ya que tiene soporte web. Está web fue embebida dentro del sistema administrativo para que sea un sistema integrado al cual el administrador puede acceder en un único punto de acceso.

5.6.5. Hosting

Debido a la estrategia que el equipo tomó, la cual consistía en ir en diferentes ocasiones al bar, con el fin de probar el producto con usuarios reales para ir validando todo el trabajo realizado y así asegurarse que se iba por el camino correcto, fue de suma importancia contar con un servicio el cual de forma adecuada y gratuita permitiera realizar de forma periódica el proceso de despliegue.

Es por esto, que para el despliegue del *backend* se escogió Heroku. El mismo consta de un servicio el cual se encarga de gestionar todo lo necesario para realizar el despliegue de la aplicación de una forma rápida y sencilla.

A su vez, Heroku permite integrarte con servicios adicionales a la infraestructura que por defecto te brinda. Para este caso, se utilizó la integración con PostgreSQL para almacenar todos los datos.

Como se menciona, el panel administrativo se encuentra deployado en Heroku, la web de toma de pedidos se encuentra deployada en *Firebase* (https://random-app-973eb.web.app/#login), además también aquí es donde se alojan las imágenes utilizadas. El *backend* del tótem también se encuentra deployado en Heroku.

Sumado a esto, la aplicación móvil ya se encuentra disponible en el Play Store ya que nuestro cliente nos indicó que el siguiente paso a probar querían que fuera la aplicación.

5.7. Principales atributos de calidad

En esta sección se detallarán los principales atributos de calidad, así como las tácticas y patrones aplicados con el fin de favorecer los mismos.

5.7.1. Usabilidad

Dentro de la usabilidad, se distingue lo intuitivo y la facilidad de uso para el usuario. Sin lugar a dudas este atributo de calidad es uno de los de mayor importancia ya que nuestro sistema tiene como fin reemplazar el modelo tradicional de atención a clientes dentro de un bar. Por lo tanto, la usabilidad dentro de la aplicación es clave para que los usuarios tengan la mejor experiencia posible.

RNF1*:* Tanto app mobile como backoffice deben ser responsive y verse bien en todos los tamaños de dispositivos.

RNF2: Debe ser usable para los clientes, cumpliendo con las heurísticas de Nielsen [53]

Una de las decisiones importantes que se tomó fue la de utilizar *Flutter* como tecnología para el desarrollo de las tres aplicaciones móviles que tiene el sistema (aplicación móvil, terminal de autoservicio y aplicación tablet cocina). *Flutter* provee elementos nativos de cada plataforma, IOS y Android, esto hace que el usuario tenga una mejor experiencia a la hora de interactuar con la aplicación, y a su vez evita que se tenga que utilizar paquetes de terceros para el diseño de la interfaz gráfica.

Como la aplicación va a ser utilizada en un lugar con muy poca luz, la misma fue diseñada en lo que se denomina *dark mode*. Esto fue testeado en el bar, simulando el mismo uso que le daría un cliente real. Comparando el *dark mode,* con el tradicional *light mode* la diferencia fue notoria, por sobre todo para la vista de las personas. El *light mode* es muy brillante para un ambiente tan oscuro como lo es el bar. Al mismo tiempo, con el fin de favorecer la accesibilidad de la aplicación, se utilizó la herramienta WAVE [54]. Esta permite evaluar la accesibilidad del *frontend* de las aplicaciones. Busca errores en el cumplimiento de las pautas definidas por la WCAG que se basa en cuatro principales principios, ser perceptible, operable, entendible y robusto.

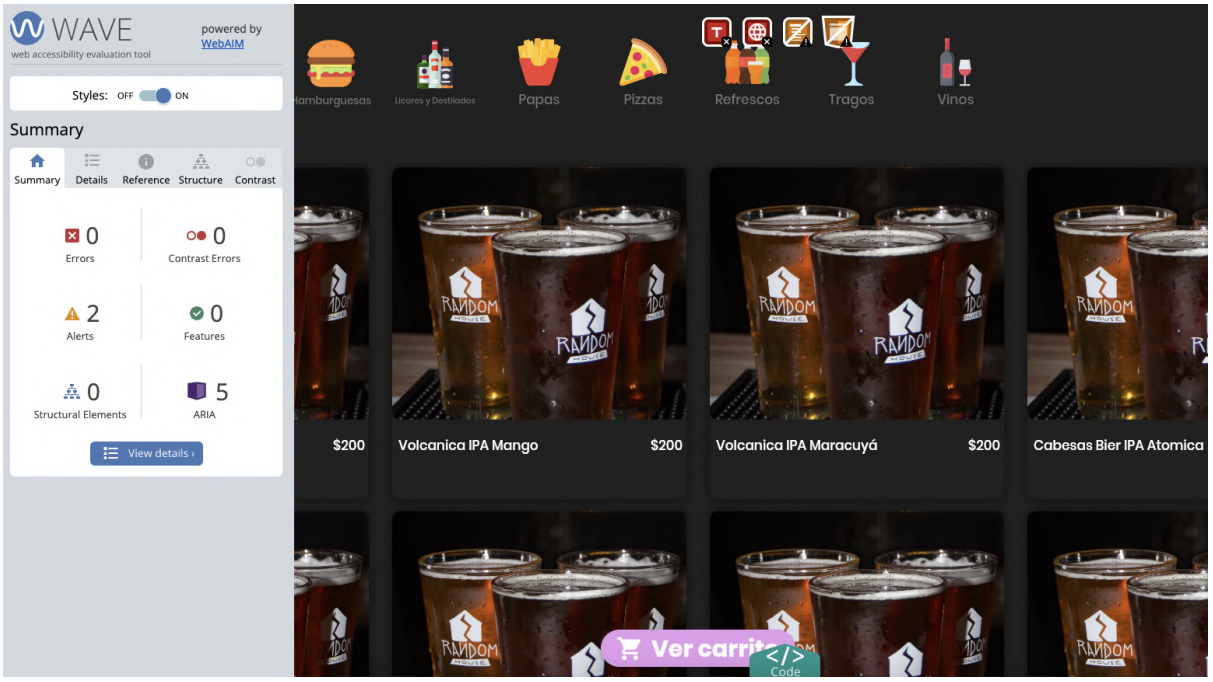


Figura 59: Herramienta WAVE

Por otro lado, para el diseño de la aplicación que se instala en la cocina para el manejo de las comandas, se optó por un enfoque *drag and drop,* esto quiere decir, que los cocineros o administradores de caja que manejen la aplicación solamente tengan que seleccionar con el dedo la comanda y arrastrarla hasta el siguiente estado. De esta forma se le brinda al administrador una experiencia más fluida sin la necesidad de tener que utilizar un teclado o un mouse.

Las heurísticas de Nielsen son reglas que nos permiten medir la usabilidad del sistema, ellas se tuvieron en cuenta en el desarrollo del producto y para evaluar durante las pruebas realizadas con usuarios. En los siguientes puntos se detallan las que se seleccionaron con mayor importancia:

- **Visibilidad del estado del sistema:** consiste en que el usuario siempre se encuentre informado sobre lo que está sucediendo en el sistema contando con el *feedback* apropiado. Para cumplir con esto se tomaron varias decisiones, entre ellas se utilizan mensajes de notificaciones dando *feedback* sobre sus acciones, cuando se realiza un nuevo pedido los productos quedan seleccionados de manera que pueda saber sencillamente que ya los incluyó, siempre puede acceder al carrito para visualizar el estado del pedido.
- **Coincidencia entre el sistema y el mundo real:** es importante que el diseño

hable el idioma del usuario, para esto todos los botones contienen textos que el usuario comprende con naturalidad. Se utiliza *Material Design* que dispone de íconos que relacionan las acciones con comportamientos en el mundo real. Además, la navegación se diseña de manera que cumple con las convenciones del mundo real, haciendo intuitivo su uso.

- **Darle control y libertad al usuario:** es normal que los usuarios realicen acciones por accidente o que se arrepientan, por lo que el sistema debe proveer siempre una forma sencilla de deshacerse dando la sensación de control y libertad. La aplicación está diseñada para poder agregar o borrar productos con facilidad o en caso de querer volver a comenzar.
- **Consistencia y estándares:** para simplificar el entendimiento de las acciones a los usuarios y no se pregunten qué significa cada una de ellas se optó por seguir convenciones de la industria utilizando *Material Design*. Es un lenguaje de diseño que permite mantener consistencia en toda la plataforma dándole claridad a través de acciones reconocidas por los usuarios.
- **Estética y diseño minimalista:** se cumple que en todas las pantallas solo se incluya información de utilidad y necesaria en cada momento.

5.7.2. Modificabilidad

La modificabilidad tiene que ver con el costo del cambio y la facilidad con que un sistema se adapta a dichos cambios.

Es un atributo de calidad al cual se le dio mucha importancia debido a que esta aplicación en un futuro se puede expandir a varios clientes. Esto permite que, a futuro, en caso de que aparezcan nuevos requerimientos, no sea complejo tener que realizar esas modificaciones y ende mantener bajo el costo de mantenimiento.

Los requerimientos no funcionales encontrados asociados a este atributo de calidad fueron:

RNF3*:* Debe ser configurable

RNF4: Fácil modificación en el desarrollo, al ser ciclo iterativo e incremental se debe poder incorporar nuevos requerimientos que vayan surgiendo o cambios. Cuanto más mantenible menos impacto.

RNF5: Seguir estándares y convenciones de codificación.

Las decisiones y tácticas utilizadas para cumplir lo anterior mencionado fueron las siguientes.

Aplicación híbrida

Al haber escogido *Flutter* como herramienta para diseñar el frontend, nos permite tener un único código centralizado el cual el mismo se despliega en dos aplicaciones distintas nativas tanto para iOS como para Android. Es por esto que *Flutter* ayuda a tener una buena modificabilidad en el sistema, al ser un único código, hace que el tiempo y corrección de errores sea mucho más reducido.

Variables de entorno

Se utilizó la táctica de *defer binding* con el fin de guardar las credenciales de servicios externos del código y guardándose como variables de entorno. Estas variables dependen del entorno en el cual se esté desplegando el código, por lo cual hace que no sea necesario modificar el código cada vez que se cambia de ambiente. Los datos de la conexión a la base de datos son un claro ejemplo de la utilización de esta táctica.

Patrón *MVC*

De forma de tener un sistema en el cual los cambios tengan el menor impacto posible, se implementó el patrón arquitectónico *MVC* para el *backend*. De esta forma queda bien definido donde debe de estar cada tipo de lógica y solo se deben modificar esos componentes a la hora de realizar un cambio.

API REST

Una *API REST* es una interfaz de programación de aplicaciones la cual permite la interacción entre las distintas aplicaciones. Todos los servicios del *backend* fueron expuestos mediante el protocolo *HTTPS* y utilizando el patrón *REST* en el cual todas las respuestas del *backend* se entregan en formato *JSON*.

Esto nos permite tener una arquitectura cliente-servidor compuesta por clientes, servidores y recursos en los cual todas las solicitudes serán gestionadas a través de *HTTPS*. Esta comunicación entre el cliente y el servidor se maneja sin estado, lo cual quiere decir que la información del cliente no se guarda en las solicitudes del lado del servidor, por lo cual cada una de ellas es independiente y se debe incluir toda la información necesaria dentro de la petición para que la misma se procese de forma correcta.

5.7.3. Seguridad

El atributo de calidad de seguridad sirve como medida para saber cuál es la habilidad del sistema para resistir usos no autorizados mientras sigue brindando sus servicios a los usuarios legítimos. En este sistema hay varios aspectos a tener en cuenta cuando hablamos de seguridad ya que la misma maneja información personal de los usuarios. Igualmente, un punto a destacar es que si bien se realizan compras con tarjetas, estos pagos son gestionados por una plataforma tercera permitiéndonos delegar la responsabilidad de mantener información sensible.

Los requerimientos funcionales asociados a la seguridad que se recabaron fueron:

RNF8: Los usuarios para acceder deben estar autenticados

RNF9: Los usuarios deberán tener roles para poder acceder a las distintas partes de la aplicación.

RNF10: La transferencia de datos entre servidores debe ser cifrada.

Las tácticas y las decisiones implementadas fueron las siguientes:

Certificado de autorización (https)

Al utilizar el protocolo de comunicación *HTTPS*, para todas las páginas del sistema, no aseguramos que la información que se envía entre los clientes y el servidor está protegida.

Autenticación del usuario

Ya sea para la aplicación web del administrador, para el uso de la aplicación móvil o

la terminal de autoservicio, todas ellas requieren de una autenticación de usuario para ingresar al sistema. Sin el uso de sus credenciales, ninguno de los distintos componentes de la solución podrá ser utilizado. Para ello se utilizó el esquema de autenticación *HTTP Bearer authentication* [55] en el cuál se utilizan tokens de seguridad llamados bearer tokens. El usuario inicia sesión a lo que el servidor le responde con el token generado por JWT [56] teniendo una validez temporal, luego cada vez que el usuario desee comunicarse con el servidor nuevamente debe incluirlo en el header para poder realizar las peticiones. En el backend se cuenta con una función middleware encargada de verificar la validez del token, si este no es válido se le responde con el código *HTTP 401 Unauthorized* y se le solicita nuevamente que se autentique.

Además de lo anterior, el sistema cuenta con la opción de iniciar sesión con las credenciales de Google, esto fue implementado con el servicio de Firebase Authentication el cual provee los estándares de la industria como OAuth 2.0 y OpenID Connect.

Encriptación de contraseña

Para asegurarnos de que cada contraseña ingresada, sea almacenada de forma seguro sin poder ser descifrada por cualquier persona la cual pueda a llegar a tener acceso a la base de datos, estas son almacenadas utilizando hashing con el método de encriptación de sha-256 y salt.

Otras consideraciones:

Api de Mercado Pago

Para todo lo relacionado a pagos electrónicos, se utiliza la *API* de Mercado Pago. La misma se encarga de todo lo relacionado a la seguridad del uso y almacenado de las tarjetas de créditos, lo cual permite desligarse por completo de esa responsabilidad.

Logs de auditoría

De manera de poder identificar fallas y posibles ataques, se optó por guardar datos de auditoría con la herramienta de Papertrail la cual permite guardar todas las

acciones y eventos en un servicio externo. De esta forma se evita sobrecargar la base de datos del *backend* y se tiene en un solo lugar centralizado todos los logs de auditoría.

5.7.4. Performance

La performance es la medida con la cual se ve la eficiencia de los recursos del sistema a la hora de ejecutar la aplicación. Cumplir con una performance óptima es de suma importancia para brindarle al usuario una buena experiencia a la hora de utilizar la aplicación. Los requerimientos funcionales que se reconocieron fueron:

RNF13: Las peticiones deben ejecutarse en menos de 3 segundos

RNF14: Las comandas deben llegar a la cocina en tiempo real

Las tácticas y decisiones utilizadas para favorecer la performance fueron:

Comunicación asincrónica

Para todas las llamadas de la API, se utilizó la programación asincrónica para así poder mejorar los tiempos de respuestas del *backend* y por ende mejorar la performance general del sistema.

Flutter

Flutter es un lenguaje el cual tiene una excelente performance la cual se compara perfectamente a la de utilizar aplicaciones nativas al sistema operativo. A continuación, se presenta una tabla comparativa con React Native, el cual era la otra opción que el equipo consideró y el cual es su mayor competidor.

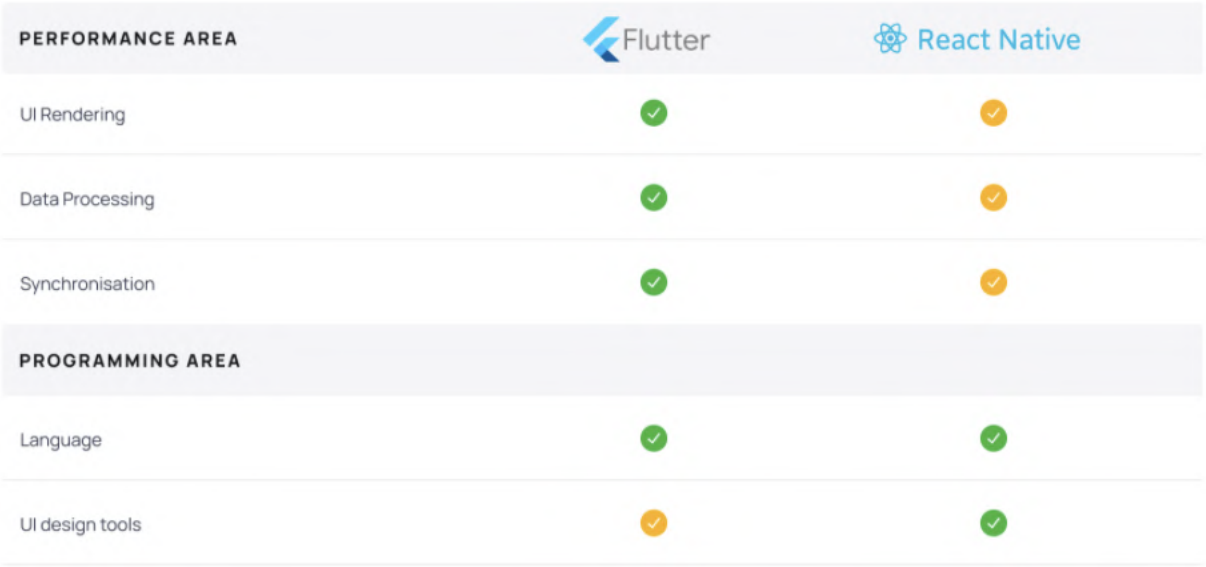


Figura 60: Flutter vs React Native

Almacenamiento de imágenes

Con el fin de no sobrecargar el sistema de *backend* con llamadas para cada vez que se tenga que cargar una imagen, se optó por utilizar el servicio de *Firebase* llamado *Firebase Storage*. De esta forma, cada vez que se van a buscar las imágenes para cargar en pantalla no hace esas peticiones al servidor de nuestro sistema, sino que lo hace al de *Firebase*.

5.7.5 Interoperabilidad

La interoperabilidad trata de la habilidad de que diversos componentes de dos o más sistemas funcionen correctamente al intercambiar información.

La forma en la cual este sistema favorece la interoperabilidad es mediante la interfaz que es expuesta por el *backend* a través de *API REST*. De esta forma, cualquier servicio externo que esté autorizado puede acceder a la información expuesta por el *backend*.

5.7.6. Portabilidad

Dado de que uno de los componentes del sistema consta de una aplicación la cual está enfocada para el público en general de bar, la misma debe de funcionar

correctamente en una gran variedad de dispositivos, sistemas operativos y distintas dimensiones de pantallas.

RNF6: Los sistemas deben correr sin problemas tanto en iOS como Android y en distintas terminales.

RNF7: El back office debe ser soportado en Chrome, Safari y Firefox.

Para solucionar este problema, se optó por una tecnología la cual permita un desarrollo multiplataforma en la cual a partir de un único código se obtuviera una aplicación funcional a distintas plataformas. Esta tecnología seleccionada para esto fue *Flutter*.

Flutter asegura portabilidad hacia una gran cantidad de distintas versiones de softwares, no solo para dispositivos *Android* (versión mínima Android 5.0) sino que también para dispositivos i*OS* (versión mínima iOS 11).

5.7.7. Disponibilidad

El atributo de calidad disponibilidad define la porción del tiempo que el sistema es funcional y trabaja. Al ser un sistema el cual se encarga manejar todo el flujo comandas del bar, es de suma importancia tener una buena disponibilidad para que el bar funcione correctamente y los clientes estén satisfechos con el servicio brindado.

Los requerimientos no funcionales detectados con respecto a la disponibilidad fueron:

RNF11: El sistema debe contar con un uptime de 90% o mayor.

RNF12: El equipo debe responder en no más de 24 horas ante fallas o errores.

Para el aseguramiento de la disponibilidad se hizo el *deploy* en Heroku el cual utiliza los servicios de *AWS,* ya que este servicio asegura un uptime de 90% o mayor [57]. Por otro lado, no solo se tomaron decisiones a nivel de servidor, sino que también se tomaron a nivel de desarrollo. Se implementaron excepciones con el fin de poder detectar comportamientos inadecuados del sistema.

Detección y prevención de fallos

Con el fin de asegurar la disponibilidad del sistema, se integró la aplicación *Bugsnag.* Es un software el cual permite el monitoreo y reporte de errores. El mismo sirve para detectar no sólo errores, sino que también *bugs* que haya en la aplicación. En caso de que se genere un error en la aplicación a la hora de ser usada en producción, estos no solo aparecerán en el *dashboard* de este software, sino que también le llegará un mail a cada uno de los integrantes reportando lo sucedido con detalle.

A continuación, se adjunta una imagen con el *dashboard* del Bugsnag:

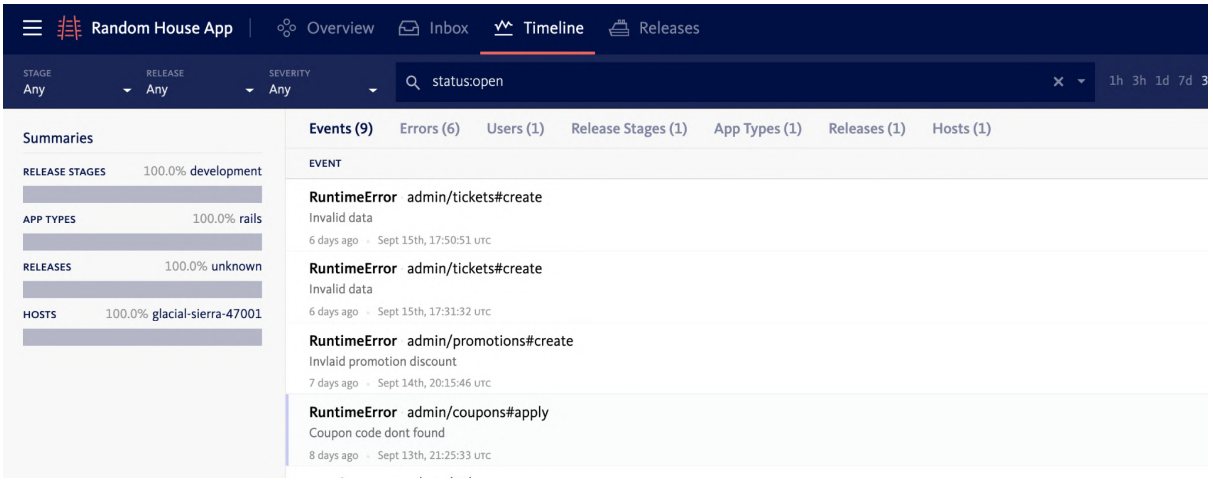


Figura 61: Bugsnag dashboard

Load balancing

Los enrutadores HTTP de Heroku distribuyen las solicitudes entrantes para su aplicación a través de sus *web dynos* [58]. Por lo tanto, escalar la capacidad de una aplicación para manejar el tráfico web implica escalar la cantidad de web dynos. Esto hace que el sistema nunca se vea saturado ya que en caso de haber múltiples peticiones simultáneamente, las mismas serán redireccionadas a distintas instancias para así poder satisfacer todas estas sin que el sistema colapse.

6. Desarrollo

6.1. Características del equipo de desarrollo

Como se mencionó anteriormente solo un miembro del equipo conocía la tecnología del *backend* por lo que el resto tuvo que capacitarse de modo tal de poder aportar en el desarrollo del mismo. Al igual que con el *backend*, solo un miembro conocía la tecnología del *frontend* por lo que los otros dos debieron capacitarse. De esta manera todos los miembros estaban capacitados para aportar en el desarrollo tanto de *frontend* como de *backend*.

6.2. Estrategias de desarrollo

Como el equipo estaba constituido por tres personas, y esa cantidad es poca para poder dividir en equipos específicos para cada área además de que el alcance del proyecto en el tiempo estipulado supuso bastante trabajo por parte de los integrantes, es que todos los integrantes desarrollaron tanto *frontend* como *backend*.

Igualmente, los integrantes que ya tenían conocimiento de las tecnologías fueron los "*go to*" en caso de que alguno de los otros tenga dudas sobre la tecnología.

El desarrollo del *backend* fue realizado en paralelo con el desarrollo del *frontend* ya que para alcanzar con el objetivo de tener un *MVP* para el primer *release* esa era la manera óptima para proceder.

En el caso de las tecnologías que ningún integrante del equipo manejaba como lo era *Firebase* y el *checkout* de Mercado Pago, estas fueron separadas para que un desarrollador se encargue de la parte de pagos y otro de la parte de *Firebase*. Ya que de por sí, son temas complejos que requieren mucha investigación previa a la integración.

El equipo utilizó la funcionalidad de *pull request* que ofrece *Github*. De esta forma, antes de que se subiera al repositorio lo que cada integrante programe, tendría que ser aprobado por los otros dos integrantes, esta modalidad se denomina *code review.*

7. Gestión de proyecto

7.1. Gestión de alcance

Dada la poca experiencia del equipo en la creación de sistemas desde cero hasta el deploy, incluyendo todas las actividades que se requieren durante el desarrollo, fue de vital importancia monitorear la velocidad del equipo durante el transcurso del proyecto para de esta forma controlar si se lograría llegar al alcance acordado inicialmente.

Para controlar esto es que al final de cada *sprint* el equipo contaba con distintas herramientas visuales para lograr discernir si se ha cumplido con lo planificado para esa instancia. De esta forma se pueden analizar los desvíos para lograr cumplir con el alcance. Se utilizaron *burndown charts* para ver cómo era el avance del proyecto (en *story points*) en base a lo ideal. Se pueden ver estas gráficas en el Anexo 12.4. Gráficas de gestión de alcance.

7.1.1. Definición de alcance

Para definir el alcance el equipo se basó en los requerimientos iniciales que tenía el cliente, luego se realizaron diferentes actividades de validación sobre los mismos como por ejemplo entrevistas, encuestas, etc. Al concluir estas actividades se logró definir el *backlog* inicial habiendo aplicado cambios a los requerimientos qué tenía el cliente en un principio.

Estos cambios fueron discutidos y explicados al cliente, llegando a la conclusión de que para el esfuerzo estimado y según lo relevado en las actividades, el alcance debería consistir en un sistema integral de gestión de comandas para el bar.

Sumado a esto, el alcance comprende a la documentación en donde se detalla todo lo referente al trabajo realizado y la puesta en producción del sistema ya que para el cliente era importante tener una prueba piloto del mismo. Una vez definido esto, se analizó para que efectivamente pudiese ser culminado con éxito dentro de los tiempos pactados. De esta manera el alcance comprendió el desarrollo, el *deploy* y la documentación del mismo.

7.1.2. Plan de release

Para gestionar el alcance el equipo decidió dividir el proyecto en tres *releases*, esta cantidad fue determinada en base a los requerimientos que abarcaría cada *release*. Se puede ver detalladamente las funcionalidades que se planificaron para cada uno de estos en el Anexo 12.8. Plan de release.

Release 1

11 de enero - 4 de abril: Para este *release* se desarrolló el primer *MVP* de forma tal que el cliente tuviera un sistema para probar y dar *feedback*.

Release 2

5 de abril - 13 de junio**:** Se trabajó sobre el primer *release* agregando nuevos requerimientos obtenidos por parte de nuestro cliente y de los usuarios que probaron el sistema.

Release 3

14 de junio - 8 de agosto**:** Nuevamente se trabajó incrementando el anterior *release* agregando los nuevos requerimientos relevados y terminando los planificado inicialmente.

7.2. Estrategia de gestión

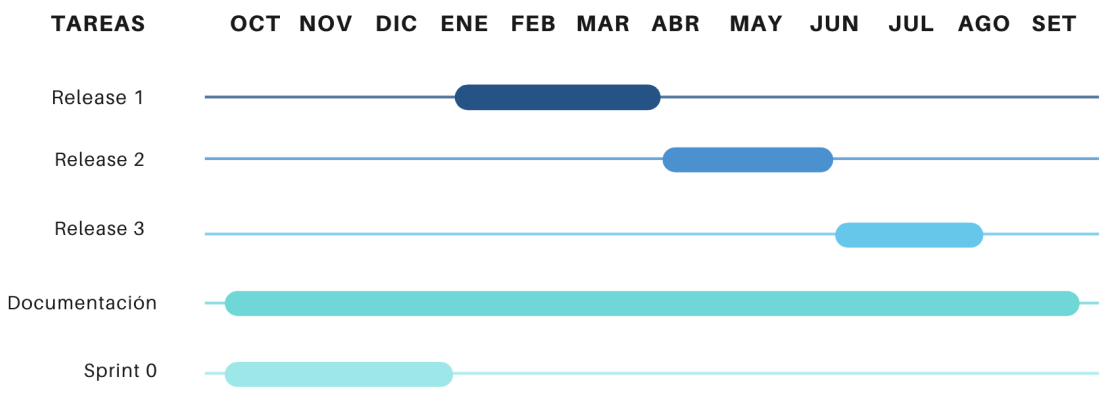


Figura 62: Cronograma

Como se puede ver, el proyecto comenzó con un *sprint* 0 cuyo objetivo era el de recabar requerimientos y conocer lo más posible sobre el producto a realizar y las tecnologías a utilizar. Al final de esta etapa se definió la arquitectura a ser utilizada y se realizaron los prototipos iniciales que serían utilizados de referencia para comenzar el desarrollo.

Luego del sprint 0 se decidió que el proyecto iba a constar de tres *releases* en donde en el primero se llegaría a un *MVP* el cual se utilizaría para ser validado.

De forma paralela se realizó la documentación teniendo como mes más importante el último.

7.3. Gestión del desarrollo

Luego del *sprint* 0 que incluye diversas actividades como la definición del *backlog,* entrevistas, encuestas, entre otras, se pasó a la fase de desarrollo que sucedió entre enero y agosto.

Para realizar este trabajo el equipo optó por utilizar *Scrum* adaptado a las necesidades. Debido a la cantidad de integrantes del equipo se tuvo que alternar las tareas asignadas a cada uno dependiendo las necesidades del momento. De todas formas, a lo largo del proyecto los integrantes trabajaron como "*full-stack*" ya que desarrollaron tanto front como *backend.*

Para la estimación de la dificultad de las *user stories* se realizó *planning poker* de forma tal que fuese un consenso de lo que pensaba cada integrante del equipo. En base a esto y a la velocidad del equipo planeada es como se definía la cantidad de *user stories* que habría en cada *release*.

Una vez que se tenían las *user stories,* cada desarrollador se asignaba una de forma tal que el resto del equipo sepa y no se trabaje sobre los mismos requerimientos.

Se priorizaron las *user stories* que afectaban directamente al flujo principal de la realización de un pedido como las más importantes.

Para tener las *user stories* ordenadas se utilizó un *backlog* el cual era gestionado utilizando la herramienta *Jira*.

7.4. Trabajo de cada sprint

A continuación, se detalla el trabajo realizado para cada *sprint*, se incluye un resumen de las tareas y eventos más importantes, métricas de velocidad en cada *sprint* y sus respectivos *retrospective highlights*.

Sprint 1

El primer *sprint* se realizó desde el 11/01/2021 al 24/01/2021.

En el mismo lo más importante fue finalizar la capacitación de las tecnologías seleccionadas, al mismo tiempo se tuvo que configurar los distintos ambientes los cuales generaron retraso.

Como consecuencia de esto en este primer *sprint* hubo un retraso de 3 *story points*.

Se crearon ambos proyectos de *frontend* y *backend* para los cuales se comenzó con las primeras funcionalidades.

En el *frontend* se creó la estructura general del proyecto y los modelos. También se desarrolló la pantalla de *login*.

Del lado del *backend* se creó la base de datos con la definición de tablas establecida y se hizo la búsqueda del *framework* de administración adecuado.

Velocidad	**Desvío por Sprint**
43	-3

Tabla 10: Sprint 1

Retrospective highlights

El equipo logró cumplir con el esfuerzo planteado.

Debido al atraso el equipo consideró que debía mejorar su velocidad, se evaluó que no fue por falta de esfuerzo, sino que se debió por usar tecnologías nuevas.

Sprint 2

El segundo *sprint* se realizó desde el 25/01/2021 al 07/03/2021.

En el frontend se desarrolló la pantalla de registro, la de restablecer contraseña y sus conexiones con el *backend* con los mismos.

En el *backend* se aplica el *framework* Active Admin y se adecua para que funcione el registro y login.

Velocidad	Desvío por Sprint
54	8

Tabla 11: Sprint 2

Retrospective highlights
El equipo logró completar el trabajo estimado y mejorar la velocidad.

Se encontró como algo a mejorar algo a mejorar el sistema usado para la gestión del proyecto (Trello) por falta de funcionalidades.

Continuaron sucediendo algunos problemas de configuración de ambientes en *Ruby on Rails*.

Se planeó para el siguiente *sprint* investigar posibles alternativas de gestión y en caso de coincidir con alguna realizar la migración necesaria. Además, sobre la comunicación del equipo se debe continuar cumpliendo con las dailys meetings.

Sprint 3

El tercer *sprint* se realizó desde el 08/02/2021 al 21/02/2021.

Se comenzó con el home screen de la aplicación y las cards de los productos.

Del lado del *backend* se hizo el *ABM* de los productos y se hizo la carga de datos de la carta del bar.

Velocidad	Desvío por Sprint

37	-9

Tabla 12: Sprint 3

Retrospective highlights

Se debería cumplir con las horas pactadas. El equipo entendió que se debía a evaluaciones de la facultad y que para los siguientes *sprints* se debe prever esto y esforzarse más si se sabe que en algunos momentos no se va a tener disponibilidad.

Se manejó la gestión correctamente y se logró solucionar los problemas con *Ruby on Rails* que había en el *sprint* anterior.

Sprint 4

El cuarto *sprint* se realizó desde el 22/02/2021 al 07/03/2021.

Dentro de la aplicación se desarrolló el bottom navigation bar que permite navegar entre las distintas tabs. Se implementó la pantalla de ver, editar el perfil y la acción de *logout*.

En el *backend* hicimos las funcionalidades para que funcionen lo desarrollado en el *frontend*.

Velocidad	**Desvío por Sprint**
42	-4

Tabla 13: Sprint 4

Retrospective highlights

Nuevamente hubo retrasos debido a exámenes y parciales más allá de que se planeó mejorar eso. Se aumentaron las horas respecto al *sprint* anterior pero igualmente no fueron suficientes. Seguimos teniendo en cuenta la necesidad de tomar medidas para mejorar los atrasos.

La comunicación con el cliente se considera buena, no surgen problemas al discutir y darnos *feedback*.

Sprint 5

El quinto *sprint* se realizó desde el 08/03/2021 al 21/03/2021.

En este *sprint* se realizó la pantalla para agregar nuevos productos, la pantalla de promociones, los modelos de las nuevas entidades y la conexiones con el *backend* correspondientes.

De igual manera, en el *backend* se adaptó el *ABM* de promociones que provee el *Active Admin*. Se comenzó con la integración de la aplicación con *Firebase*.

Velocidad	Desvío por Sprint
44	-2

Tabla 14: Sprint 5

Retrospective highlights
El equipo se esforzó más de lo pactado trabajando más horas. A pesar de esto existieron problemas por faltas de conocimientos sobre la tecnología al realizar la integración con *Firebase*, concluyendo en un desvío negativo. Este problema lo enfrentó un integrante del equipo solo y de haberlo comunicado probablemente esto no hubiera exigido tanto esfuerzo.

Se plantea una mejor comunicación dentro del equipo, en cada *daily* es importante dejar en claro si alguien se encuentra atascado con algo.

Sprint 6

El sexto *sprint* se realizó desde el 22/03/2021 al 04/04/2021.

Al finalizar este *sprint* se culmina el primer *release* y se concluye en el *MVP*.

Se deja funcionando correctamente las pantallas de gestión de la carta en el *backoffice web*.

Se inicia con el desarrollo del carrito de compras en la aplicación.

Se realizó el *deploy* del *backend* en *Heroku*. Desarrollo de pantalla de selección de *beeper*.

Velocidad	Desvío por Sprint
54	8

Tabla 15: Sprint 6

Retrospective highlights
Gracias al *release* se pudo recabar nuevos requerimientos y mejoras.
Se mantuvo una reunión con el experto en UI/UX Martin Solari.
Como aspecto a mejorar se destaca subir los cambios de desarrollo con más regularidad.

Sprint 7

El séptimo *sprint* se realizó desde el 05/04/2021 al 18/04/2021.
Se incorpora el slider horizontal de categorías en la pantalla principal. Se continúa desarrollando el carrito y se agregan notas a las comandas. Se comenzó a desarrollar la pantalla para la gestión de comandas, tanto el *frontend*, como la comunicación con el *backend*.

Velocidad	Desvío por Sprint
50	4

Tabla 16: Sprint 7

Retrospective highlights
Se tuvo la reunión con Amalia Álvarez la cual fue muy fructífera además se trabajó a buen ritmo. Se debe cuidar la gestión de incidentes y ser consistente con la metodología pactada.

Sprint 8

El octavo *sprint* se realizó desde el 19/04/2021 al 02/05/2021.

Se continúa trabajando en el desarrollo del carrito, se le agrega la funcionalidad de poder modificar las cantidades del producto agregado y la posibilidad de eliminarlo. Por otro lado, se desarrolla la funcionalidad de que se apliquen de forma automática las promociones a los productos. Al mismo tiempo, se desarrolló para que desde el backoffice se pueda agregar una promoción.

Velocidad	Desvío por Sprint
48	2

Tabla 17: Sprint 8

Retrospective highlights
Se trabajó a un buen ritmo, se logró el objetivo de recuperar el tiempo perdido en los últimos dos *sprints*. Se debe prestar más atención a los *pull request*.

Sprint 9

El noveno *sprint* se realizó desde el 03/05/2021 al 16/05/2021. Se implementa el control de stock para todos los productos del bar. También, para el backoffice, se agrega la funcionalidad de crear cupones generales o para usuarios específicos.

Velocidad	Desvío por Sprint
46	0

Tabla 18: Sprint 9

Retrospective highlights
Se trabajó al ritmo esperado y como resultado no hubo desvíos en el *sprint*. Se le prestó más atención a los *pull requests*. No se realizaron pruebas unitarias de las funcionalidades del *backend* desarrolladas debido a que se priorizo terminar las historias de usuario del *sprint*.

Sprint 10

El décimo *sprint* se realizó desde el 17/05/2021 al 30/05/2021. Se implementó la posibilidad de poder pagar en efectivo. De la mano con esto, se agregó una vista en el backoffice en la cual se puedan visualizar todas las órdenes pagas en efectivo. Se

hizo el desarrollo correspondiente para las *push notifications* tanto para los cupones como para la notificación de que una comanda está pronta.

Velocidad	Desvío por Sprint
47	1

Tabla 19: Sprint 10

Retrospective highlights
La velocidad del equipo fue la esperada. Se cumplió con las historias de usuario pactadas. Mejorar la comunicación con el cliente.

Sprint 11

El onceavo *sprint* se realizó desde el 31/05/2021 al 13/06/2021. Se desarrolló un *dashboard* en el backoffice con información relevante para el cliente. Se crea el proyecto web para así poder acceder desde una pestaña del panel administrativo.

Velocidad	Desvío por Sprint
48	2

Tabla 20: Sprint 11

Retrospective highlights
Se logró tener la velocidad acordada y por lo tanto se cumplió con todas las historias de usuarios. Se logró llegar con los requerimientos acordados para el segundo *release*. Se mejoró la comunicación con el cliente ya que era crucial previo al lanzamiento de un *release*. Mejorar la priorización de actividades.

Sprint 12

El doceavo *sprint* se realizó desde el 14/06/2021 al 27/06/2021. Se realizaron cambios en la UI recabados en el segundo *release*. También se realizaron cambios en la UX tales como cambiar la pantalla de agregar un producto por un *bottom modal sheet*. Se siguió con el desarrollo de la aplicación web.

Velocidad	Desvío por Sprint

47	1

Tabla 21: Sprint 12

Retrospective highlights

Se debe prestar más atención a la hora de registrar las horas trabajadas. Se cumplió con las historias de usuarios pactadas.

Sprint 13

El treceavo *sprint* se realizó desde el 28/06/2021 al 11/07/2021. Se comenzó la integración con Mercado Pago para poder utilizarlo a la hora de pagar una orden.

Velocidad	**Desvío por Sprint**
42	-4

Tabla 22: Sprint 13

Retrospective highlights

Debido a exámenes y parciales no se logró cumplir con la cantidad de historias de usuarios pactadas.

Sprint 14

El catorceavo *sprint* se realizó desde el 12/07/2021 al 25/07/2021. Se continuó con la integración de mercado pago y se desarrolló la posibilidad de visualizar los pedidos no pagos para cada usuario de la app. Por otro lado, también se agregó la funcionalidad de poder repetir una orden. Se realiza el deploy al Google Play Store.

Velocidad	**Desvío por Sprint**
41	-5

Tabla 23: Sprint 14

Retrospective highlights

No se llegó nuevamente a la cantidad de historias de usuario pactadas por la falta de tiempo de los integrantes.

Sprint 15

El quinceavo *sprint* se realizó desde el 26/07/2021 al 08/08/2021. Se realiza *deploy* la aplicación web en *Firebase*. Se agrega la opción de historial de pedidos para la aplicación para usuarios, se actualiza el menú con la carta actualizada del bar.

Se integra la web de pedidos con el *backoffice*.

Se pone en producción todo el proyecto para la utilización del mismo en el bar.

Velocidad	Desvío por Sprint
53	7

Tabla 24: Sprint 15

Retrospective highlights
Se logró recuperar todo lo que se había atrasado en los últimos dos *sprint* para así poder cumplir con todas las historias de usuarios pactadas.

7.5. Gestión de riesgos

7.5.1. Identificación de riesgos

El primer paso en la gestión de riesgos era el identificarlos lo antes posible para que el equipo sepa a que potencialmente se podría enfrentar. Esto es importante ya que todas las decisiones que se tomaron fueron teniendo en cuenta los riesgos que se habían identificado en el momento. Se puede ver la tabla en detalle en el Anexo 12.12. Tabla de riesgos.

Para mayor especificidad el equipo separó los riesgos en 4 categorías: tecnológicos, del equipo, del producto, académicos.

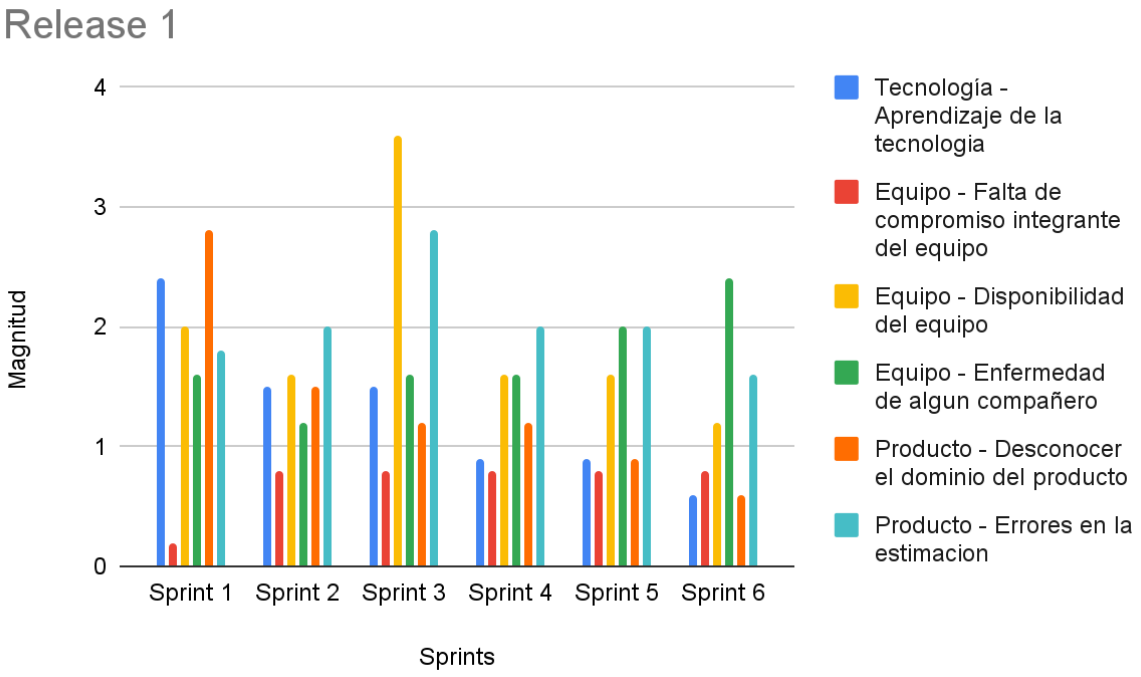


Figura 63: Identificación de riesgos Release 1

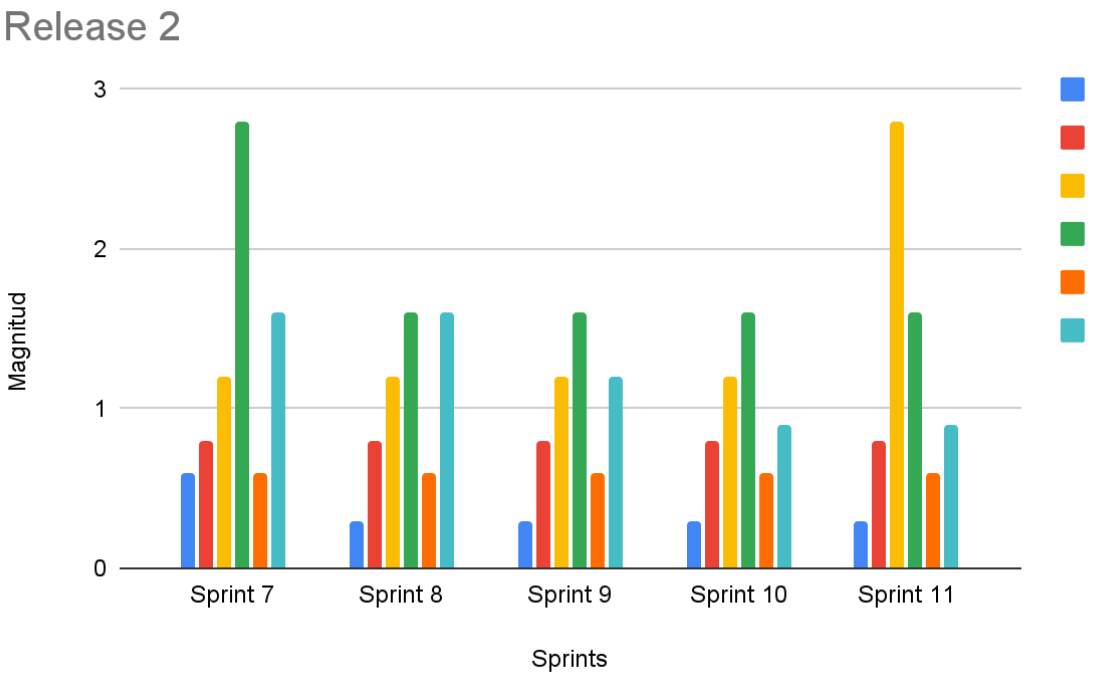


Figura 64: Identificación de riesgos Release 2

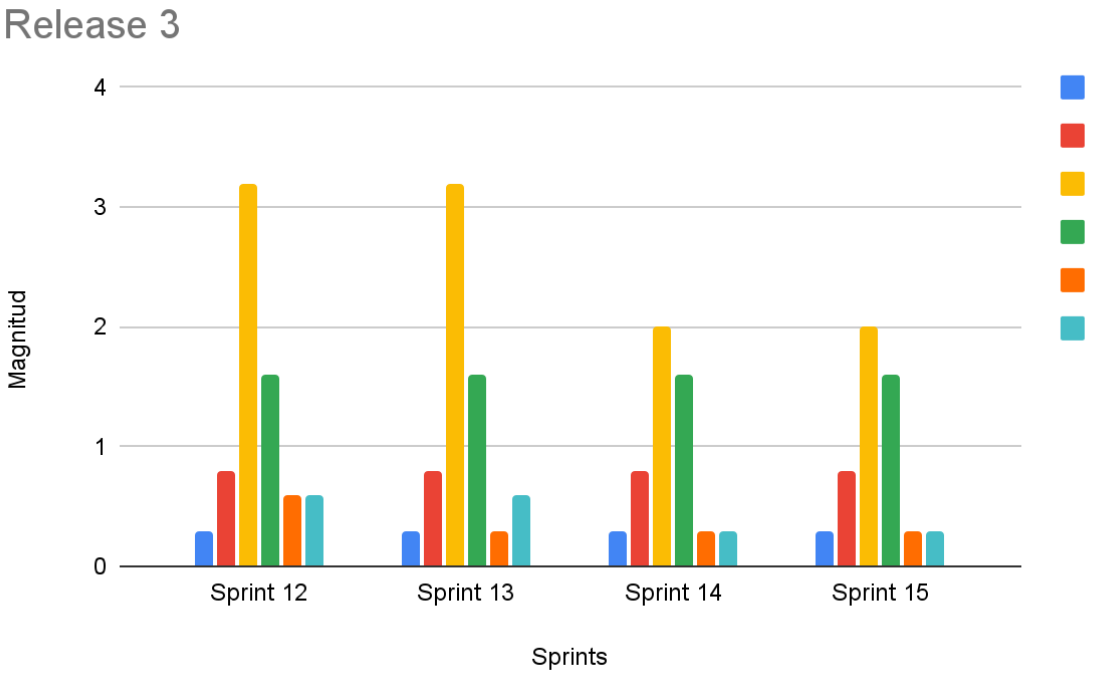


Figura 65: Identificación de riesgos Release 3

7.5.2. Asignación de prioridad

Para tener un mayor control sobre los riesgos identificados el equipo optó por asignar distintas prioridades a cada uno de estos.

Una vez identificado un riesgo el equipo le asigna probabilidad (entre 0 y 1) e impacto (entre 1 y 5), esto es, la probabilidad que se estima de que el riesgo se materialice y el impacto que tendría sobre los objetivos en el caso de que se materialice. De esta manera se llega a la magnitud, métrica la cual sirve al equipo para priorizar los riesgos y saber a cuáles prestar más atención.

		Release 1					
Categoria	Nombre	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Tecnologicos	Tecnología - Aprendizaje de la tecnologia	2.4	1.5	1.5	0.9	0.9	0.6
	Equipo - Falta de compromiso integrante del equipo	0.2	0.8	0.8	0.8	0.8	0.8
Equipo	Equipo - Disponibilidad del equipo	2	1.6	3.6	1.6	1.6	1.2
	Equipo - Enfermedad de algun compañero	1.6	1.2	1.6	1.6	2	2.4
Producto	Producto - Desconocer el dominio del producto	2.8	1.5	1.2	1.2	0.9	0.6
	Producto - Errores en la estimacion	1.8	2	2.8	2	2	1.6

Release 2					Release 3			
Sprint 7	Sprint 8	Sprint 9	Sprint 10	Sprint 11	Sprint 12	Sprint 13	Sprint 14	Sprint 15
0.6	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
1.2	1.2	1.2	1.2	2.8	3.2	3.2	2	2
2.8	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6
0.6	0.6	0.6	0.6	0.6	0.6	0.3	0.3	0.3
1.6	1.6	1.2	0.9	0.9	0.6	0.6	0.3	0.3

Tabla 25: Asignación de prioridad.

7.5.3. Comportamiento de los riesgos

Los riesgos fueron reevaluados al final de cada *sprint* en donde se calculaba nuevamente la probabilidad y el impacto. Esto era necesario ya que algún riesgo puede variar en alguna de estas variables y tener más chance de materializarse.

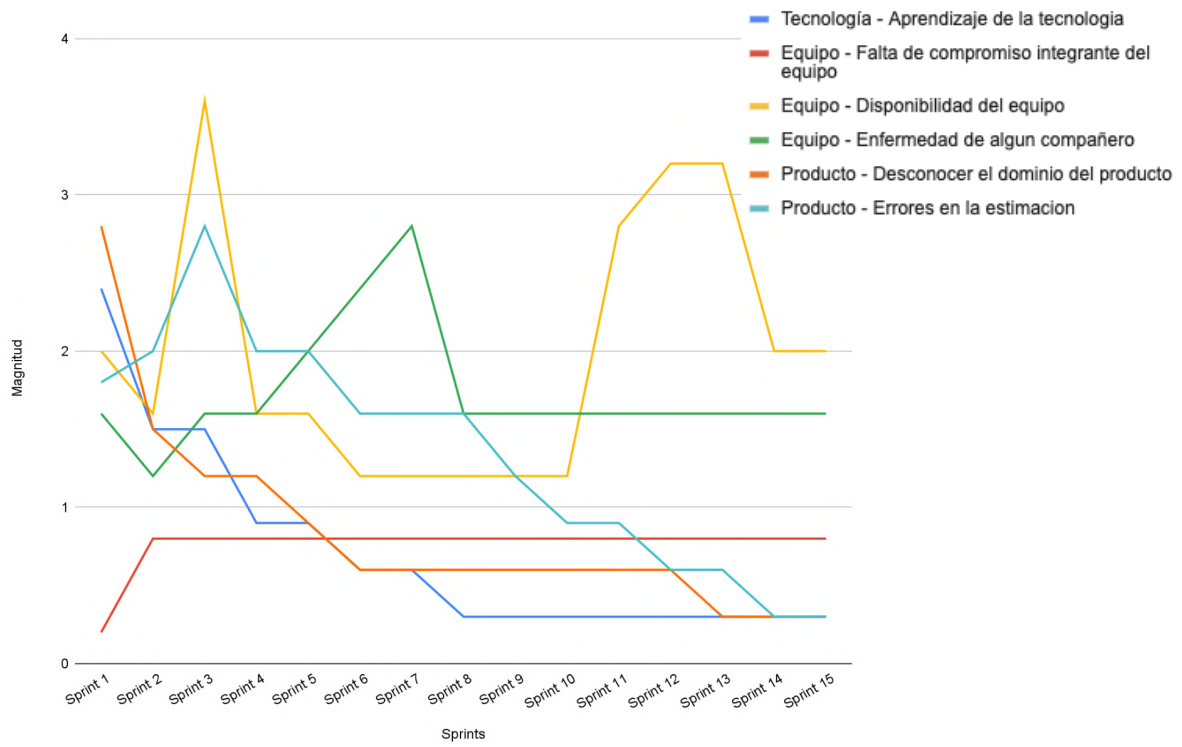


Figura 66: Comportamiento de los riesgos

En la gráfica se puede ver la evolución de los riesgos a lo largo del desarrollo del proyecto.

- Aprendizaje de la tecnología

Como se puede observar en la gráfica, el riesgo de que los integrantes del equipo tengan problemas en lo que respecta al aprendizaje de la tecnología era alto en un principio y esto se debe a que la mayoría no conocía la tecnología por lo tanto tenía más chance de tener problemas en el aprendizaje.

Se puede ver cómo a lo largo de los *sprints* la magnitud de este riesgo disminuye, esto se debe al progreso en el aprendizaje de los integrantes con las tecnologías.

- Disponibilidad del equipo

Como se puede observar, el riesgo de que la disponibilidad del equipo se vea afectada es bastante variable. Esto se debe mayoritariamente a los periodos de exámenes/parciales ya que en estas épocas los integrantes solían estar menos disponibles.

- Enfermedad de algún compañero

En cuanto a este riesgo hay que hacer principalmente dos apreciaciones.

Debido al contexto actual, desde un principio se valoró con bastante probabilidad el hecho de que algún integrante se enferme. Por lo tanto, quizás en otro contexto la magnitud de este riesgo podría haber sido menor, pero el equipo consideró que debido a la incertidumbre del progreso de la pandemia se debía estar más pendiente.

Como lo refleja la gráfica, durante el transcurso del proyecto los casos diarios de COVID-19 fueron cambiando. Particularmente cuando sube abruptamente la magnitud de este riesgo coincide con la suba de casos diarios de COVID-19 y también con el hecho de que un compañero tuvo que realizar cuarentena debido a tener un contacto con un positivo.

- Desconocer el dominio de producto

Como se puede ver en la gráfica, el riesgo de desconocer el dominio del producto en un principio era alto ya que ninguno de los integrantes del equipo había trabajado en nada relacionado con el rubro gastronómico, por lo tanto, lo desconocíamos

completamente. Es por eso que la probabilidad era alta y a lo largo del proyecto, como lo refleja la gráfica, fue disminuyendo a medida que el equipo fue tomando conocimiento del dominio.

7.5.4. Materialización de riesgos

De todos los riesgos mencionados anteriormente solamente dos de ellos se materializaron o tomaron suficiente trascendencia como para prestarles atención y poder afectar directamente al transcurso normal del desarrollo.
Por lo cual se tuvo que tomar medidas al respecto, siguiendo el plan definido y por lo tanto no repercutan negativamente a largo plazo en el proyecto.

- Disponibilidad del equipo: dado a que los integrantes del equipo no solo se encontraban trabajando en este proyecto, sino que también estaban cursando materias en la universidad y con sus respectivos trabajos, hubo ocasiones en las cuales este riesgo se vio materializado. Para que estos no tengan mayor incidencia a largo plazo el equipo tenía como plan de respuesta aumentar las horas de desarrollo posteriores a la materialización del riesgo.
- Enfermedad de algún compañero: si bien no se materializó al 100%, debido al contexto actual de pandemia hubo momentos en los cuales este riesgo tomó un valor elevado ya que sucedió que integrantes del equipo tuvieran que realizar cuarentena.

En la siguiente tabla se pueden ver los distintos planes de respuesta y contingencia para cada uno de los riesgos definidos.

Riesgo	Plan de respuesta	Plan de contingencia
Tecnológicos		
Aprendizaje de la tecnología	Recurrir a personas con experiencia	Utilizar la documentación oficial
Riesgo plugin *Flutter*	Utilizar una solución alternativa	Utilizar plugins bien calificados
Equipo		

Falta de compromiso integrante del equipo	Tener una reunión grupal y hablar con el integrante	El gerente del proyecto controla las horas invertidas semanalmente
Mala relación con el cliente	Tener una reunión con el cliente para evaluar la relación	Tener reuniones regularmente y estar siempre en contacto con el cliente
Enfermedad de algún compañero	Incrementar las horas de trabajo del resto de los integrantes para llegar a los objetivos planteados	Tratar de no exponerse de forma innecesarias en actividades multitudinarias
Disponibilidad del equipo	Incrementar las horas de trabajo del resto de los integrantes para llegar a los objetivos planteados	Tener planificadas con tiempo las tareas a realizar para que los integrantes del equipo se puedan organizar
Existencia de problemas entre integrantes del equipo	Reunión entre los integrantes del equipo para solucionar los problemas	Cada miembro del equipo debe tener claro sus obligaciones y responsabilidades dentro del grupo
Producto		
Desconocer el dominio del producto	Reunirse con el cliente para tener una mejor explicación de aquellos temas que no se entendieron	Tener reuniones frecuentes con el cliente y realizar distintas actividades para comprender el dominio a fondo
Errores en la estimación	Ver las causas para corregir el error y reevaluar el alcance	Analizar regularmente las métricas obtenidas y evaluar si las estimaciones siguen siendo correctas
Abandono del cliente	Buscar otro bar de las	Mantener una buena

	mismas características que le interese nuestra solución	comunicación mostrando avances, manteniendo al cliente motivado
Académico		
Abandono del tutor	Pedir a la Universidad ORT que se asigne otro tutor al equipo	El equipo debe cumplir con las responsabilidades esperadas

Tabla 26: Plan de contingencia y respuesta de riesgos

7.6. Métricas de gestión

7.6.1. Burndown chart

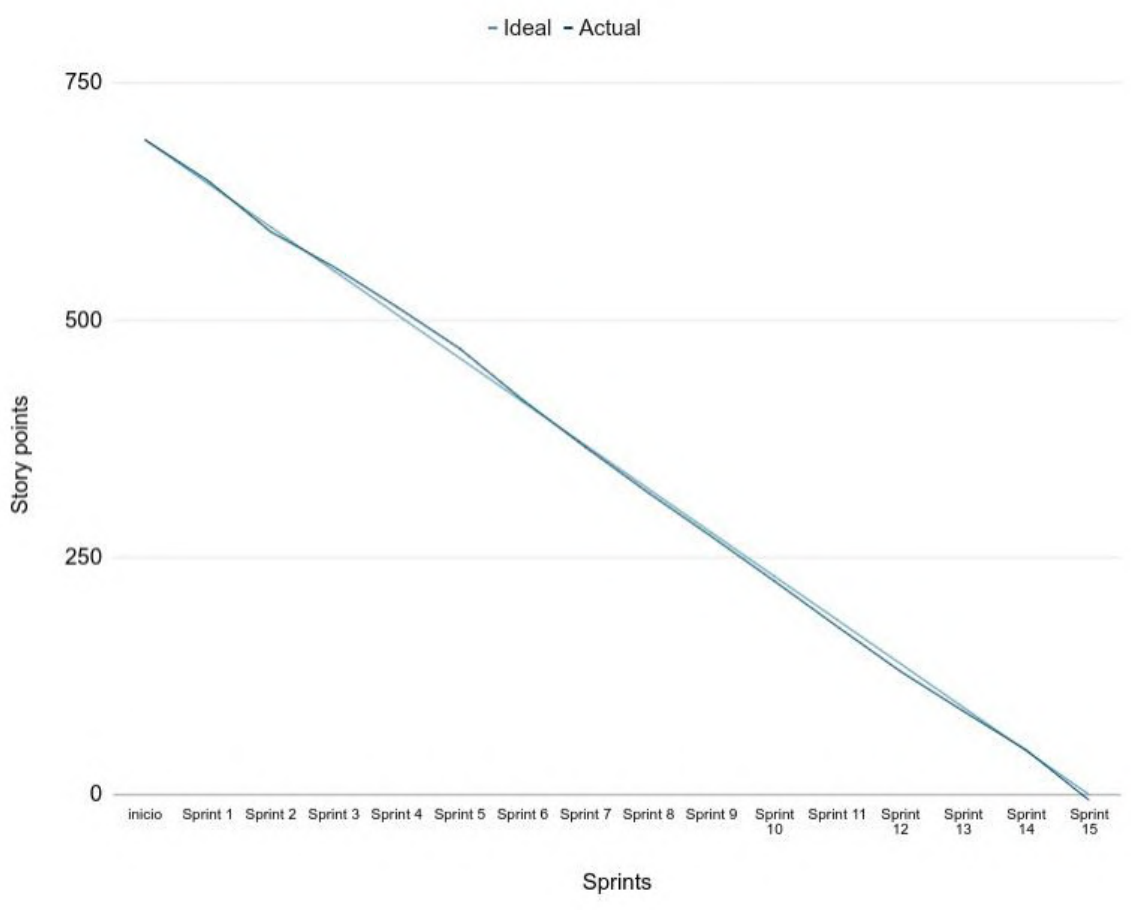


Figura 67: Burndown Chart

Durante todo el proyecto se utilizó el artefacto *burndown chart* para llevar un seguimiento de los *story points* restantes y los completados hasta el momento.

De esta manera el equipo tenía una herramienta visual para poder comprender fácilmente el estado del proyecto y de esta manera poder tomar decisiones en cuanto a ajustar el esfuerzo o el alcance.

Se pueden ver leves desvíos respecto a la velocidad ideal los cuales se explican en los siguientes capítulos.

7.6.2. Velocidad por sprint

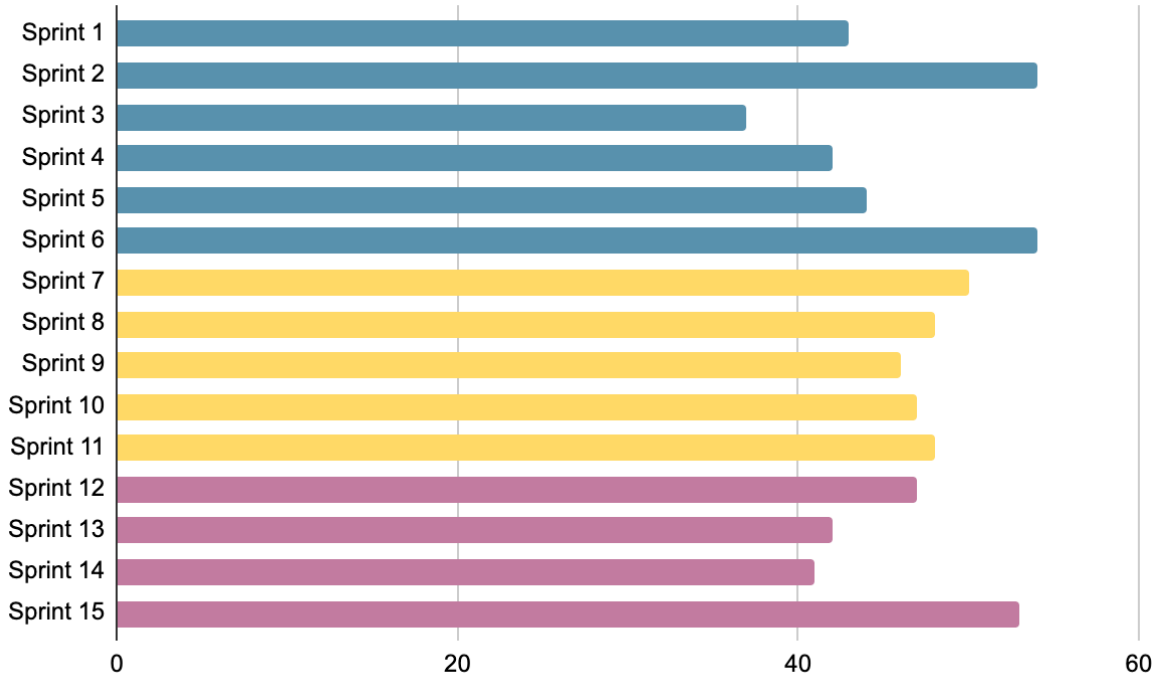


Figura 68: Velocidad por sprint

En la gráfica que se presenta se puede ver la velocidad del equipo en cada *sprint* en *story points*. La velocidad estimada inicialmente era de 46 *story points* por *sprint*.

Como se puede ver en la gráfica el equipo tuvo variaciones en la velocidad a lo largo del proyecto, el mayor causante en las variaciones de velocidad del equipo fueron las evaluaciones (tanto parciales como exámenes) que los integrantes tenían a lo largo del semestre. Es por eso que en ciertos *sprints* en específico el equipo se enlentece. Otro factor importante en lo que a la velocidad respecta es la expertise

para/con la tecnología, a medida que transcurría el tiempo y el equipo iba tomando más experiencia con la tecnología, la velocidad aumento. Por lo que la experiencia afectó a la velocidad ya que cada vez menos se tenía que recurrir a herramientas como Google para entender cómo hacer una tarea o saber cómo solucionar un problema. En el último *sprint* se puede ver una aceleración respecto a los anteriores, y eso se debió a que un objetivo muy importante para el equipo era la puesta en producción. Para que esto se haga realidad el equipo decidió aumentar la velocidad.

7.6.3. Desvío por sprint

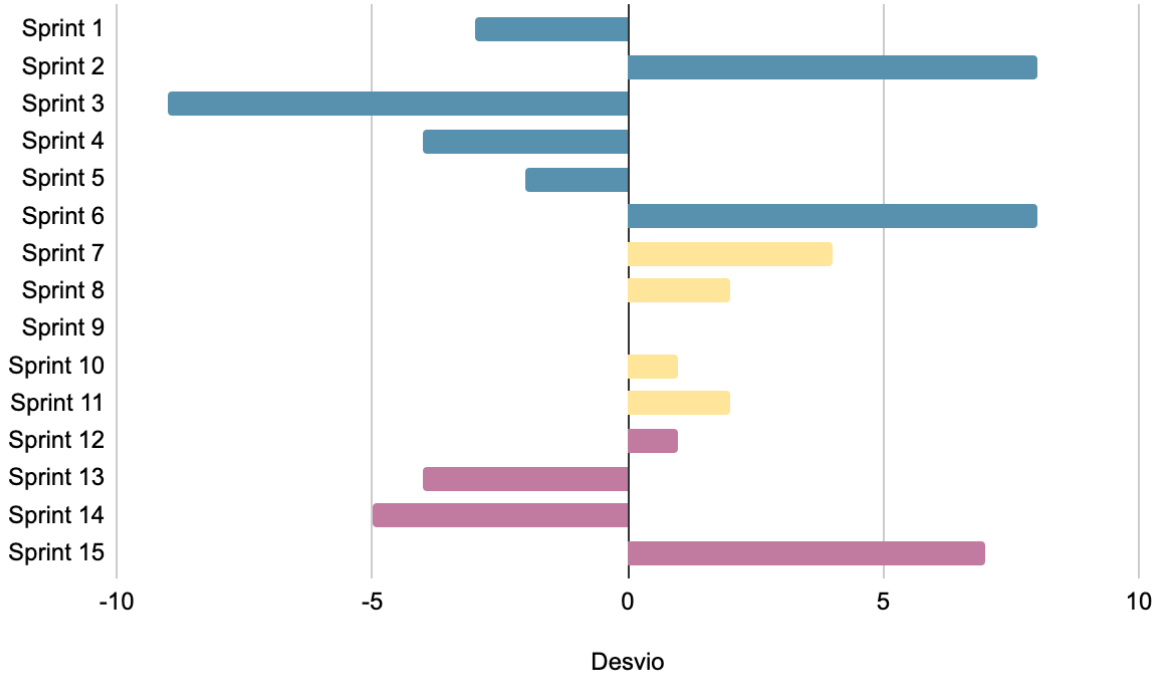


Figura 69: Desvío por sprint

Lo explicado anteriormente se puede ver reflejado en la gráfica anterior en donde se puede ver el desvío de *story points* por *sprint*.

Al analizar la gráfica se puede ver que en el primer *sprint* no se logró cumplir con la cantidad de *story points* estimada inicialmente, eso se debió principalmente por desconocimiento de la productividad de cada integrante y también por comenzar con un proyecto nuevo con tecnologías nuevas. Se puede ver como en el siguiente *sprint* se logra compensar el desvío inicial y hasta hacer más *story points* de lo

previsto, esto también se debió a que el equipo sabía que en los siguientes tres *sprints* coincide con época de parciales y exámenes por lo tanto iba a disminuir la cantidad de tiempo disponible. Es por esto que se puede ver reflejado en los *sprints* 3,4 y 5 que el equipo se atrasó. En el *sprint* número 6 y *sprint* final del *release* se puede observar como el equipo aceleró la velocidad y compenso el desvío previo.

En el segundo *release* se puede ver como el equipo luego de haber estado trabajando un *release* juntos pudo mantener un buen ritmo sin tener ningún desvío negativo.

En el tercer y último *release* sucedió que en el *sprint* 12 se continuó con la velocidad que se venía llevando en los anteriores *sprints* pero en el 13 y 14 nuevamente hubo un desvío negativo ya que coincide con fecha de parciales y exámenes. Algo a tomar en cuenta del equipo es que se subestimó la disponibilidad en período de evaluaciones. A pesar de que se sabía que había eventos importantes se siguió manteniendo el estimado inicial ya que se pensaba reponer las horas perdidas poniendo más esfuerzo por parte de cada uno, lo cual no sucedió de la forma que se esperaba.

En el último *sprint* del último *release* se puede ver como el equipo nuevamente acelera y se logra aumentar mucho la velocidad para de esta manera lograr llegar al *release* con los requerimientos inicialmente pactados.

Si bien a lo largo del proyecto ocurrieron desvíos el equipo se planteó como objetivo que estos se acerquen a cero al final de cada *release,* para que estos comprendan todas las funcionalidades planeadas. Basándose en las *burndown charts* se logró visualizar esto, aumentar el esfuerzo y lograr el objetivo de cada *release*.

7.6.4. Distribución del esfuerzo por área de trabajo

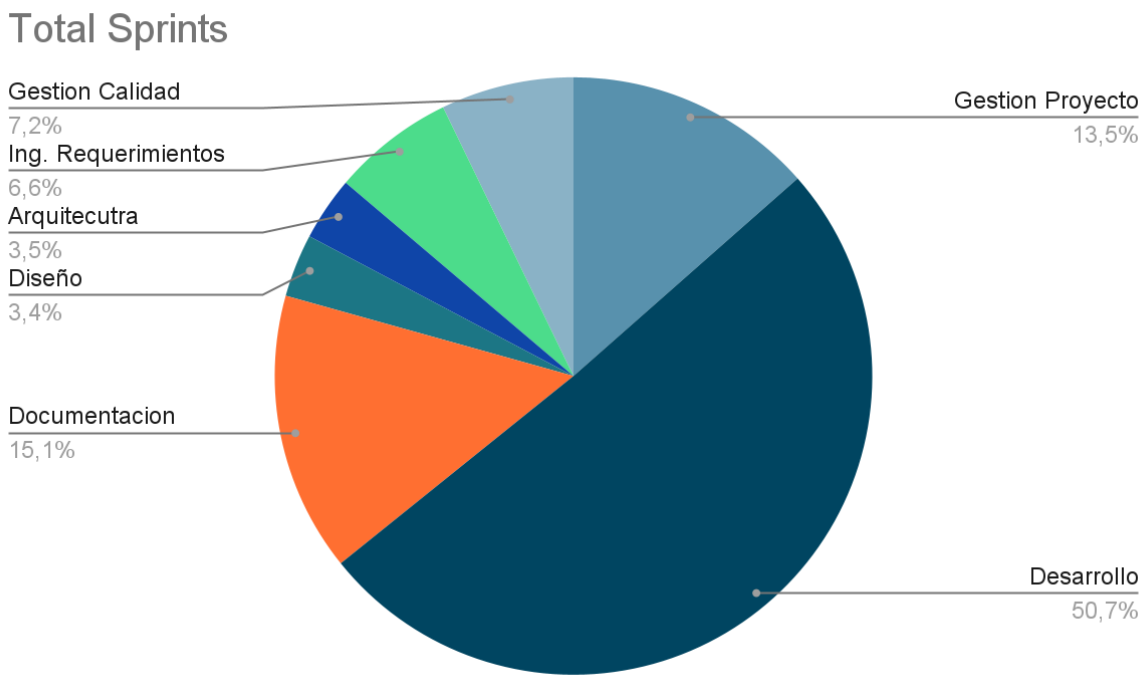


Figura 70: Distribución del esfuerzo por área de trabajo

En esta gráfica también se encuentra todo el esfuerzo referente al *sprint 0* dentro de cada área correspondiente. Como se puede observar, el mayor esfuerzo estuvo concentrado en el desarrollo. Lo cual tiene sentido ya que el proyecto cuenta con muchos flujos y pantallas distintas según los roles de los usuarios. Es por esto que el desarrollo supuso más del 50% del esfuerzo del equipo. La siguiente área en esfuerzo fue la de gestión del proyecto, lo cual también tiene sentido ya que se invirtió mucho tiempo en lo que respecta a la correcta aplicación de *Scrum*, esto implica tener *daily meetings*, *sprint retrospectives*, entre otras reuniones. El resto del esfuerzo se encuentra distribuido entre la gestión de calidad, ingeniería de requerimientos, arquitectura, diseño y documentación.

La gestión de calidad supuso un esfuerzo considerable ya que se le dio mucha importancia el asegurar la misma. Se plantearon distintos objetivos para asegurar la calidad no solo de producto sino también de proceso. Se definió un plan en el cual fue dividido por fases y en cada una de ellas se realizaron distintas actividades.

Si bien la ingeniería de requerimientos se realizó en su mayoría en un principio, luego durante el desarrollo también se recabaron requerimientos, como por ejemplo en los *releases*.

El esfuerzo para la arquitectura se realizó al principio del proyecto ya que los requerimientos base estaban claros y en principio no aparecerán nuevos de forma tal que haya que cambiar la arquitectura. La definición de la arquitectura fue hecha por el equipo y luego nos asesoramos con el experto en el asunto Gastón Mousques.

El diseño fue realizado en un principio, pero fue evolucionando a lo largo del proyecto en base a *feedback* que indicaron los usuarios en los *releases* así como también el cliente.

La documentación implicó gran parte del esfuerzo realizado ya que el equipo fue documentando a lo largo del proyecto y prácticamente el último mes se dedicó en su mayoría a documentación.

7.7. Herramientas de apoyo para la gestión

7.7.1. Gestión y comunicación

- WhatsApp

Esta herramienta se utilizó como método de comunicación principal ya que facilita la comunicación del equipo permitiendo enviar imágenes, videos y archivos. Como todos los miembros del equipo trabajan esta herramienta es útil ya que si alguno tiene algo para decir lo envía y cuando el resto está disponible lo contesta. Por medio de esta aplicación se realizaron las *daily meetings* del equipo. También se gestionaron por este medio las reuniones con el tutor y con el cliente.

- Google Drive

Esta herramienta se utilizó como nube en donde el equipo guardaba y compartía todos los archivos referentes al proyecto.

- Google Forms

Esta herramienta se utilizó para la realización de encuestas.

- Google Slides

Esta herramienta se utilizó para la realización de las presentaciones para cada revisión y defensa.

- Toggl

Esta herramienta se utilizó para llevar un registro de horas trabajadas por cada estudiante. Se puede ver el registro de estas en el Anexo 12.13. Horas promedio trabajadas.

- Microsoft Teams

Esta herramienta se utilizó para tener reuniones virtuales tanto con el tutor como con los expertos.

7.7.2. Gestión de versionado

- GitHub

Esta herramienta se utilizó como *SCM* en donde se crearon dos repositorios, uno para el *backend* y uno para el *frontend*

- Trello

En un principio comenzamos utilizando esta herramienta como apoyo de *scrum*. Teníamos columnas en donde se encontraban: *Product Backlog*, *To do*, *Doing*, *Done*. De esta manera era sencillo para los miembros del equipo poder asignarse tareas y no trabajar en requerimientos que está trabajando otro integrante. Con el transcurso de los *sprints* el equipo se dió cuenta que esta herramienta tenía carencias que podrían hacer el trabajo de gestión más sencillo y por eso se migró a *Jira*.

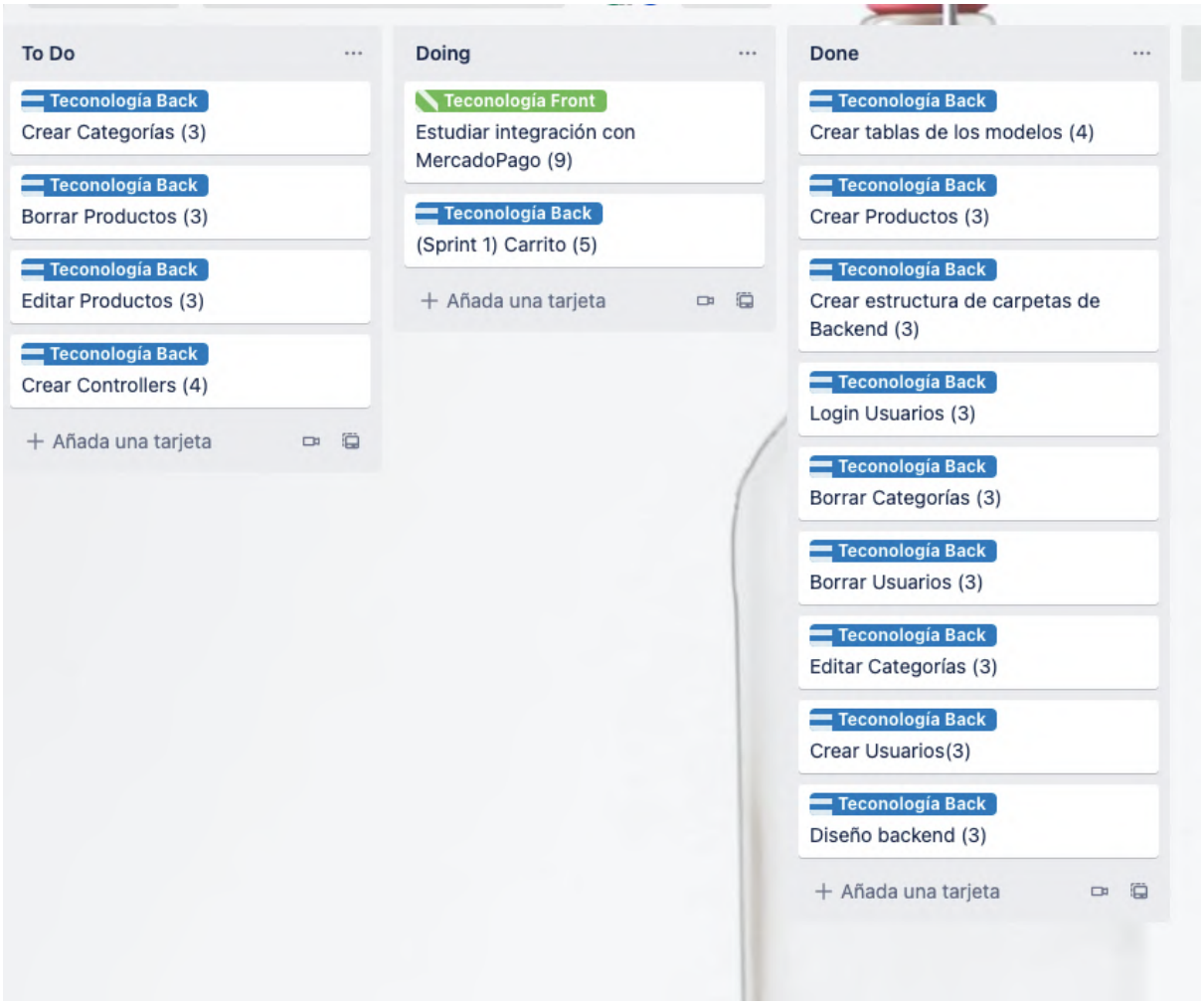


Figura 71: Trello

- Jira

Esta herramienta no se utilizó desde un principio como se comentó anteriormente, sino que se migró luego de concluir que era mejor (para el proyecto) que Trello. Las mejoras que ofrece son mayoritariamente para poder utilizar *Scrum* de una forma más sencilla y con herramientas y *addons* para una mejor gestión del proyecto.

- Microsoft Word

Esta herramienta se utilizó para la gestión de la documentación.

7.7.3. Desarrollo

- Visual Studio Code

Esta herramienta se utilizó para desarrollar tanto *frontend* como *backend*.

- Figma

Esta herramienta se utilizó para la realización de prototipos y así poder mostrarle al cliente el flujo para poder discutirlo.

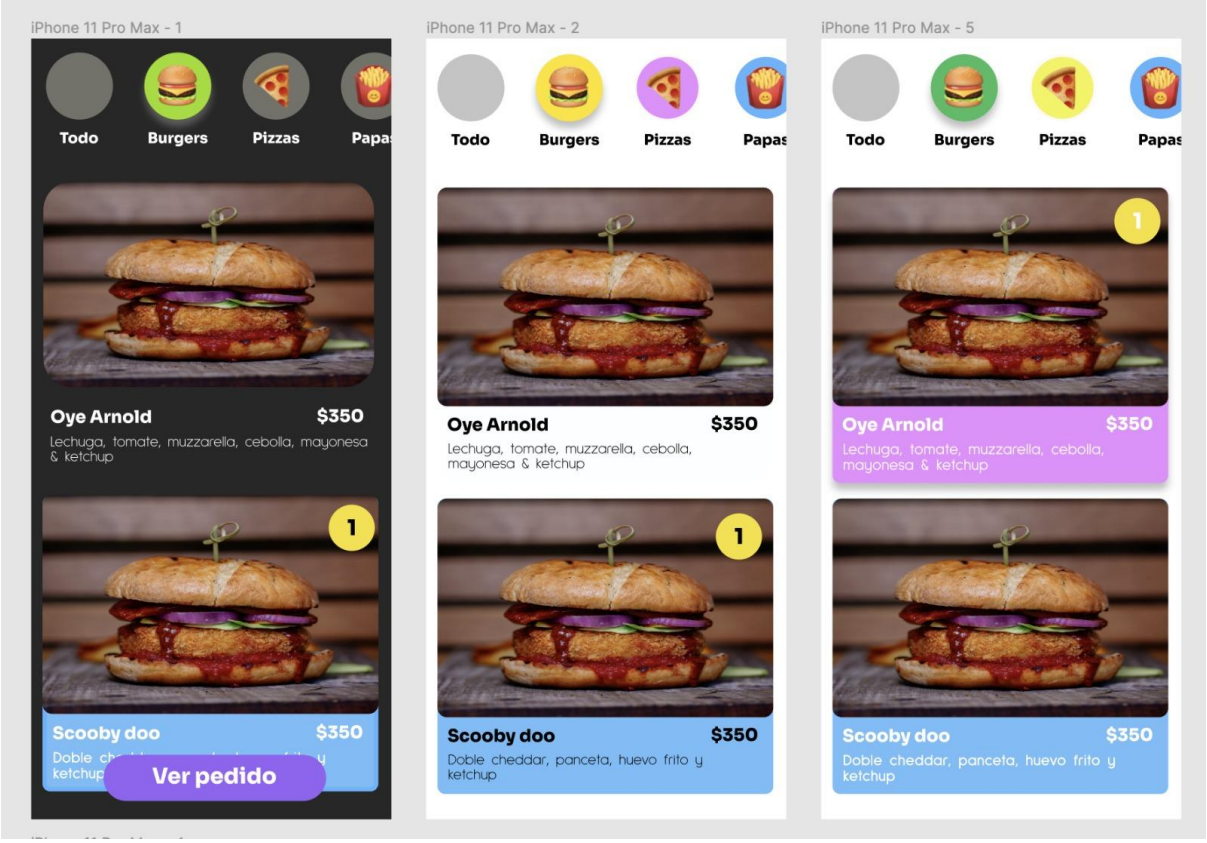


Figura 72: Figma

- PgAdmin

Esta herramienta se utilizó para poder acceder a la base de datos y poder visualizar las tablas y los datos de una manera sencilla lo que es muy conveniente durante el desarrollo.

8. Gestión de calidad

El objetivo de este capítulo es el de presentar todas las decisiones, acciones y procesos realizados que involucren tanto la calidad del producto como del proceso a lo largo del proyecto.

Este es un software que se desarrolló con el fin de ser puesto en producción y que los clientes del bar lo utilizarían regularmente para realizar sus pedidos. Es por esto, que es de suma importancia poder asegurar la calidad no solo del producto, sino que también del proceso.

8.1. Objetivos de calidad

Con el fin de poder asegurar la calidad, fueron establecidos ciertos objetivos los cuales tuvieron que ser cumplidos por el equipo a lo largo de todo el proyecto.

8.1.1. Producto

Para el objetivo de calidad del producto, se definió cumplir con todos los requerimientos funcionales y no funcionales. Al mismo tiempo, se definió cumplir con un código de buena calidad para así por último poder validar el producto con el fin de verificar que efectivamente satisfaga las necesidades del cliente y la de los usuarios finales

8.1.2. Proceso

Con el fin de aumentar la eficiencia, poder analizar resultados y disminuir fallos, se definieron distintas actividades para el proceso. A lo largo de todo el proyecto, se iban analizando los resultados obtenidos y al mismo tiempo se iba dejando registro de ellos. Las métricas sacadas a partir de esto se pueden encontrar en la sección 7.5. Métricas de gestión.

8.2. Plan de Calidad

El equipo definió un plan de calidad, en el cual se definen de un conjunto de actividades las cuales pertenecen a una fase distinta del proyecto. De esta forma,

todo el equipo está alineado bajo un mismo criterio y se cumpliría con los objetivos de calidad del proyecto.

Cabe destacar, que estos elementos fueron establecidos antes de haber comenzado el desarrollo para acompañar al equipo a lo largo del proyecto. A continuación, en la siguiente tabla se presentan las actividades del plan para cada fase.

Fase	Actividad	Producto Resultado	Producto/s Consumido/s
Sprint 0	Estudio del problema	Problema del proyecto	Reuniones con el cliente, Investigación sobre el rubro. Estudio de flujo del problema.
	Análisis de los resultados	Resultado de entrevistas, encuestas, reuniones con clientes	Reuniones con expertos en el tema, entrevistas y encuestas con posibles usuarios finales.
	Buscar soluciones	Solución de proyecto	Ingeniería inversa.
	Definición de tecnologías a utilizar	Justificación de tecnología a utilizar	Artículos técnicos y comparativos sobre las tecnologías de front y back
	Estudio de las herramientas tecnológicas	Documento de investigación tecnológica	Información de diferentes tecnologías, necesidades del cliente
	Análisis de la competencia	Comparación entre distintos competidores	Investigación en la web, comunicación con competidores, pruebas de otros productos de competencia

Ingeniería de requerimientos	Recabar requerimientos	Documento de requerimientos	Entrevistas con cliente, análisis del flujo de las comandas, encuestas a potenciales usuarios, observaciones, ingeniería inversa
	Especificación de requerimientos	ESRE	Requerimientos, actores, necesidades del cliente
	Validación de ESRE	ESRE validado	ESRE
	Priorizar	Product Backlog	Entrevista con cliente, con potenciales usuarios, observaciones, encuestas e ingeniería inversa
Diseño	Diseño de la arquitectura	Documento de arquitectura	Diagrama de arquitectura, análisis de soluciones similares, atributos de calidad
	Revisión de la arquitectura	Documento de arquitectura revisado	Documento de arquitectura
	Análisis de requerimientos no funcionales	Atributos de calidad	Requerimientos no funcionales
Construcción	Codificación de la funcionalida	Fuentes	ESRE, planificación de iteración

	d de una iteración		
	Revision de codigo	Código revisado	Pull request, revisión de a pares
	Sprint review	Mejoras en los procesos	*Sprint* backlog, código fuente
	Sprint retrospectiv e	Mejoras en los procesos	*Sprint* backlog, código fuente
	Implementa ción de pruebas unitarias	Pruebas unitarias codificadas	Product backlog,
Testing	Diseño de casos de prueba	Planilla de casos de prueba	ESRE
	Pruebas de usabilidad	Reporte de pruebas de usabilidad	Pruebas de usabilidad / Prototipos
	Pruebas de portabilidad	Reporte con lista de incidentes	Jira, Plan de pruebas
	Verificación con testing funcional	Reportes de ejecución de testing	Aplicación, Plan de pruebas
	Validación con pruebas con usuarios	Pruebas con usuarios y prototipos	Aplicación

Tabla 27: Actividades del plan de calidad.

8.3 Aseguramiento de calidad

Una vez el equipo tuvo definido el Plan de Calidad, se elaboró un Plan de Aseguramiento de Calidad para así poder cumplir con los objetivos planteados. Se tuvieron en cuenta un conjunto de estándares con el fin de generar un impacto positivo en la calidad del proyecto tanto para el proceso como para el producto. Al

mismo tiempo, le permite al equipo reducir la cantidad de errores y en caso de que aparezcan que todos sepan cómo corregir esos defectos.

8.3.1. Uso del inglés

Todo el código fue desarrollado en el idioma inglés, si en algún momento es necesario incorporar nuevos desarrolladores al equipo, la comprensión del código no será un problema ya que el inglés es el idioma estándar para el desarrollo.

8.3.2. Estándares de código

Debido a que en el proyecto se manejan códigos con dos lenguajes diferentes, fue necesario aplicar ciertos estándares para obtener un código mantenible. A su vez, el equipo siguió varios de los estándares que se plantean en el libro *Clean Code de Robert C. Martin* [59].
Específicamente, los estándares seguidos para cada tecnología fueron:

Tecnología	Estándar
Flutter	*Flutter: Best practices and tips* [60]
Ruby on Rails	*Rails Development: Coding Conventions & Best Practices* [61]

Tabla 28: Estándares de código.

8.3.3. Estándares de documentación

La Universidad ORT Uruguay provee de ciertos estándares que espera que todos los proyectos de final de carrera cumplan. Estos mismos fueron los que el equipo siguió para el desarrollo de la documentación. Los estándares aplicados fueron:

Documento	Descripción
302-Fi	Normas específicas para la presentación de trabajos finales de Carrera de la Facultad de Ingeniería.

303	Hoja de verificación de formato de trabajos finales de carreras de la facultad de ingeniería.
304	Normas para el desarrollo de trabajos finales de carrera.
306	Orientación para títulos, resúmenes o abstracts e informes de corrección de trabajos finales de carrera.

Tabla 29: Estándares de documentación

8.3.4. Estándares para control de versionado y repositorios

El equipo decidió que el *backend* y el *frontend* de la aplicación estuvieran cada uno en un repositorio separado. Para ambos proyectos se utilizó *Github*. Por otra parte, para el manejo de la documentación a lo largo de todo el proyecto se utilizó Google Drive.

8.3.5. Revisiones

Las revisiones fueron de vital importancia a lo largo de todo el proyecto para poder detectar oportunidades de mejora en el producto y en el proceso.

Para las revisiones de código, el equipo utilizó la funcionalidad de *pull request* que ofrece *Github*. De esta forma, antes de que se subiera al repositorio lo que cada integrante programe, tendría que ser aprobado por los otros dos integrantes, esta modalidad se denomina *code review.*

Por otro lado, se realizaron distintas revisiones, algunas de estas fueron con el cliente, las cuales se hacían como parte de *sprints reviews* y también al final de cada *release* se hace una revisión más grande yendo a probar el producto que se tenía en ese momento al bar. A su vez, también se realizaron otros tipos de revisiones en las cuales distintos expertos de la universidad hacían comentarios sobre el proyecto.

De estas revisiones con expertos, la primera fue con Gastón Mosques, quien le brindó al equipo un asesoramiento en cuanto al área de arquitectura de *software*. Por otro lado, el equipo tuvo una reunión con Martín Solari para la revisión de usabilidad. Junto con Martin, se llegó a la conclusión que la usabilidad es uno de los

atributos de calidad más importante dentro del proyecto, por lo que aportó mucho y se tuvo especial énfasis en la misma. Por último, pero no menos importante, el equipo tuvo una revisión con Amalia Álvarez, para relevar las actividades que se realizaron para el aseguramiento de calidad del proyecto.

Finalmente, el equipo cumplió con las tres revisiones formales que la Universidad ORT fija a lo largo del proyecto. Las misma constan en presentar los avances que el equipo realizaba a lo largo del tiempo y se recibe una devolución la cual sirve para detectar áreas a mejorar.

Estas revisiones fueron primero con Leonardo Scafarelli, luego con Rafael Bentancur y por último con Amalia Álvarez. Para todas estas se realizaron informes los cuales se encuentran en el Anexo 12.1. Informes de revisiones

Por último, una vez puesto el piloto en marcha en el bar, se le realizó al cliente una entrevista de satisfacción. La misma constó en una serie de preguntas y respuestas con el fin de poder evaluar el trabajo que se realizó desde la perspectiva del cliente. La misma se puede ver en el Anexo 12.11. Entrevista de satisfacción del cliente.

8.3.6. Validación

El equipo consideró crucial que las validaciones sean realizadas por potenciales usuarios, por lo que se mantuvieron varias instancias de evaluación con potenciales usuarios finales, tanto como en la etapa de ingeniería de requerimientos como luego durante el desarrollo.

8.3.7. Pruebas

8.3.7.1. Pruebas unitarias

Una de las formas en la cual el equipo se aseguró que estaba construyendo un *software* de buena calidad fue mediante la implementación de pruebas unitarias sobre todas las funcionalidades del *backend*. Esto implicó el desarrollo de pruebas unitarias para cada una de las historias de usuarios relacionadas al *backend* que se encontraban en cada *sprint*.

Ruby on Rails tiene la ventaja de que al momento de crear el proyecto por defecto ya crea un directorio específico en el cual se pueden implementar todas las pruebas del sistema. El equipo aprovechó esta funcionalidad de *rails* y por lo tanto todas las pruebas fueron realizadas allí sin la necesidad de utilizar un *framework* de terceros. Antes de subir el código al servidor, las funcionalidades nuevas programadas tenían que haber pasado todas sus pruebas unitarias correspondientes para así de esta forma poder reducir riesgos de incidentes y al mismo tiempo mejorar la calidad del código.

8.3.7.2. Pruebas funcionales

Con el fin de asegurarse que todas las diferentes pantallas del *frontend* iban funcionando de forma correcta a medida que se iban implementando, el equipo optó por hacer pruebas funcionales periódicamente a todo el *frontend* del sistema. De esta forma, en conjunto con los otros tipos de pruebas, se aseguró el sistema en su completitud (*frontend* y *backend*) estuviese probado.

A continuación, se deja un ejemplo de un caso de prueba funcional realizado:

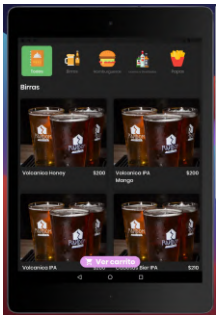
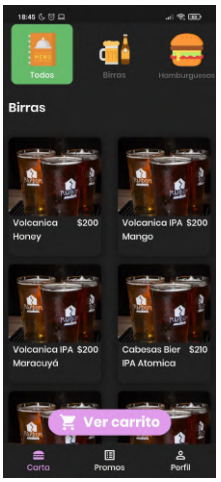
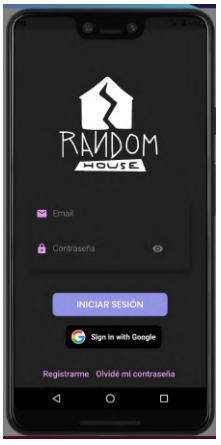
Identificador	CP32.2
User Story asociada	US32
Descripción	Agregar producto a carrito
Pasos	Ingresar a la aplicación Seleccionar un producto del carrito Seleccionar cantidad de ese producto a agregar Click en ‘Agregar al carrito’
Resultado esperado	El producto se agrega al carrito correctamente

Tabla 30: Ejemplo de caso de prueba funcional.

8.3.7.3. Pruebas de portabilidad

Dado que uno de los componentes del sistema consta de una aplicación móvil enfocada para el público general del bar, se realizaron distintas pruebas sobre

diferentes dispositivos de iOS y Android en diferentes resoluciones. Por otra parte, también se probó en distintas tablets la terminal autoservicio para asegurarse que no solo funcionara correctamente en un dispositivo. A continuación, se deja evidencia de estas pruebas:

Dispositivo	Aplicación	Sistema operativo	Tamaño pantalla	Captura
Nexus 9	Terminal Autoservicio (Flutter)	Android	8,9”	
Pocophone x3 nfc	Aplicación móvil (Flutter)	Android	6,67”	
Pixel 3 XL	Aplicación móvil (Flutter)	Android	6,3”	

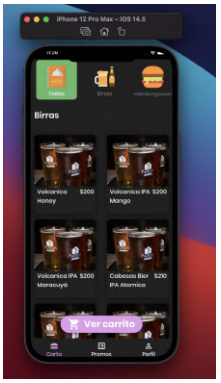
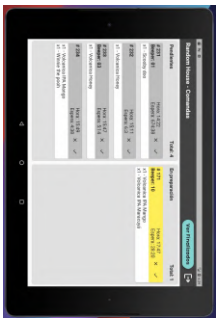
iPhone 12 Pro Max	Aplicación móvil (Flutter)	iOS	6,7''	
Nexus 9	Aplicación comandas cocina (Flutter)	Android	8,9''	

Tabla 31: Pruebas de portabilidad.

8.3.8. Satisfacción del cliente

Fue de suma importancia la buena comunicación que hubo con el cliente a lo largo de todo el proyecto. A través del grupo de WhatsApp, y también mediante las conversaciones que se mantuvieron en el bar, se iba recibiendo retroalimentación continua del cliente.

Esto fue crucial para llegar a un producto final en el cual el cliente se sintiera satisfecho con el mismo, era muy importante que el cliente se sintiera parte de este desarrollo y pudiera ir opinando a lo largo de todo el proceso sobre posibles mejoras y nuevas funcionalidades.

Fue así como una vez finalizado el tercer *release* y puesto en marcha el piloto, el cliente sintió que el software desarrollado por el equipo le aportaba valor a su negocio, ya sea tanto para brindarle una mejor experiencia a los clientes del bar como también para mejorar todo lo vinculado a la gestión de los pedidos.

Una vez finalizado el proyecto, se le realizó una entrevista al cliente con el fin de evaluar el trabajo realizado por el equipo desde su perspectiva.

La entrevista se puede ver en el Anexo 12.11. Entrevista de satisfacción del cliente.

8.4. Gestión de incidentes

Como parte de la gestión de calidad se estableció un flujo el cual se debía cumplir para asegurar la correcta gestión de los incidentes surgentes. Antes de explicar este flujo cabe destacar que se utilizó *Jira* como herramienta para realizar un correcto seguimiento de los incidentes.

En primer lugar, cada vez que se detectaba un nuevo incidente el mismo se registraba en *Jira* en el *product backlog*. Una vez registrado se pasaba a la asignación de prioridad del mismo. Para poder tener un control de los mismos, el equipo decidió separar a los incidentes en tres niveles de prioridad, estas prioridades eran bajo, medio y alto.

Una vez asignada esta prioridad se pasa a solucionar el incidente, era importante que se moviera la tarjeta a la columna *Doing* en *Jira* para que el resto de los integrantes estuvieran al tanto de que se estaba trabajando sobre ese incidente. Una vez solucionado, se realizaban las pruebas correspondientes para asegurarse que no se vuelva a repetir el incidente.

Por último, una vez fuera testeado, y asegurándose de que pasara todas las pruebas, el incidente era movido en *Jira* a la columna *Done* para que el resto del equipo supiera que él mismo fue resuelto. En caso de no pasar alguna prueba, el incidente volvía al *backlog* con los comentarios correspondientes.

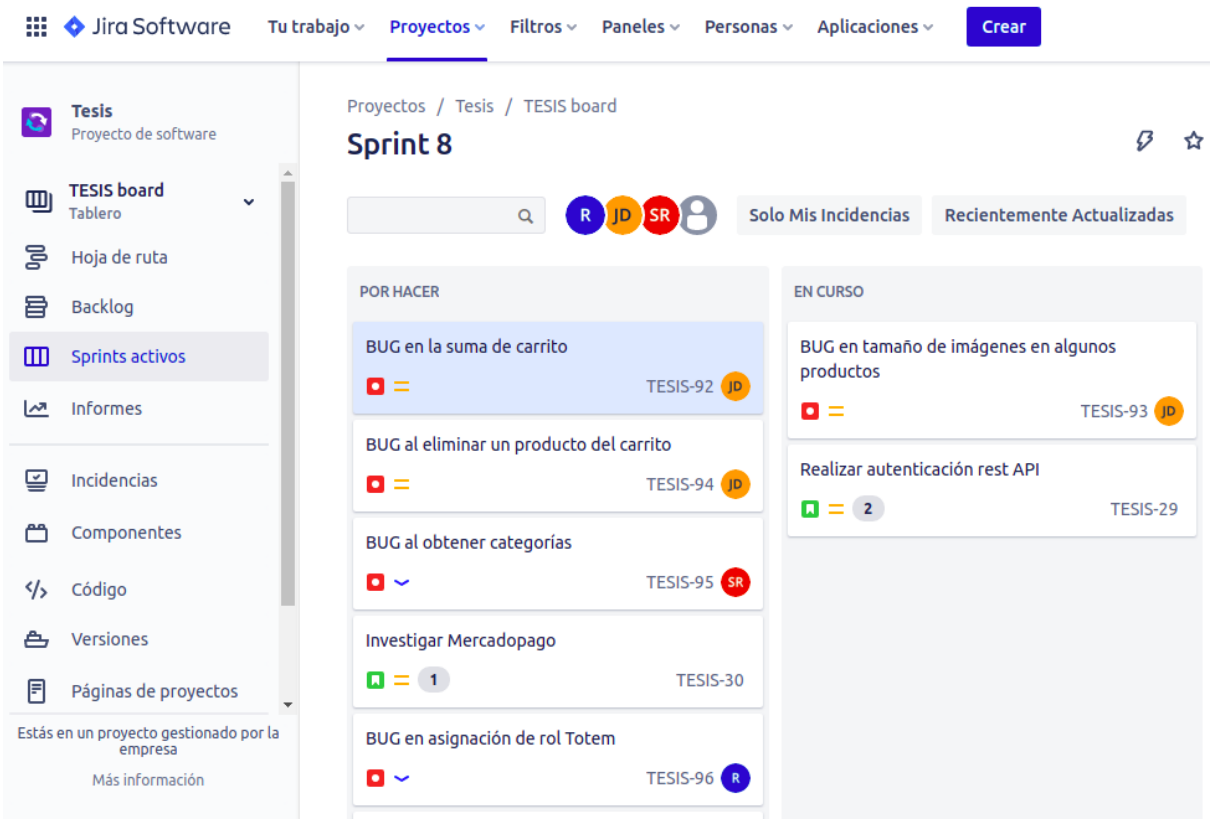


Figura 73: Jira

8.5. Métricas

A continuación, se detallarán las diferentes métricas utilizadas. Las mismas fueron claves y se iban analizando *sprint* a *sprint* con el fin de ir tomando decisiones las cuales hicieron que se cumpla con los objetivos previamente establecidos

8.5.1. Métricas del proceso

Estas son aquellas medidas las cuales de forma cuantitativa le permitieron al equipo obtener una visión de la eficacia del proceso de *software*. De esta forma, el equipo podía ajustar el proceso a lo largo del proyecto para alcanzar los objetivos.

Métricas de gestión de incidentes

Si bien depende de las *user stories* incluidas en cada iteración, esta métrica sirvió para encontrar un valor en donde el equipo se sentía cómodo y podía continuar el flujo de trabajo, incluyendo las nuevas historias de usuario en cada *sprint*, y a su vez, arreglando los defectos encontrados. En otras palabras, el equipo prestó

especial atención a esta métrica para anticiparse y organizarse, previendo determinada cantidad de tiempo para arreglar los defectos encontrados al final de cada *sprint*. También sirvió como motivación para intentar evitar la mayor cantidad de defectos posibles.

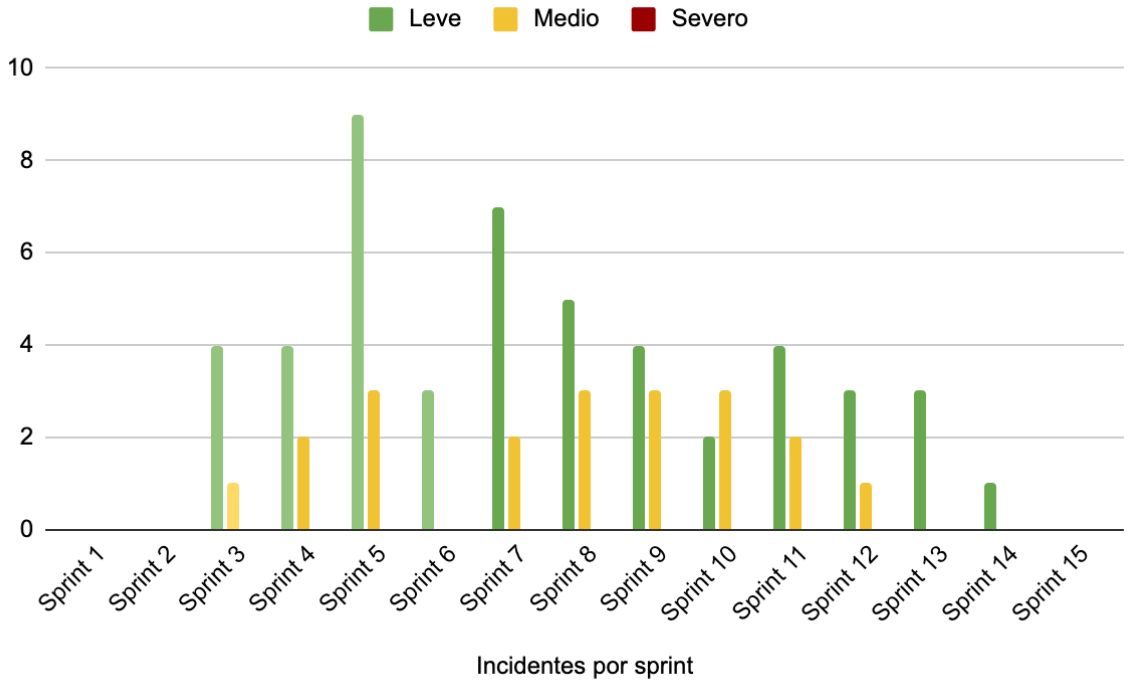


Figura 74: Cantidad de incidentes identificados por sprint

8.5.2. Métricas del producto

Estas métricas son las que nos permiten monitorizar un producto para ir determinando a lo largo de todo el proyecto la calidad del mismo.

Índice de mantenibilidad

Esta es una métrica, la cual como su nombre lo indica, permite calcular cuán mantenible es el software. En otras palabras, calcula a partir de un valor que tan fácil es ejecutar las tareas de gestión de mantenimiento sobre el software.

Para esto, la herramienta realiza un cálculo a partir de otras métricas que devuelve un valor entre 0 y 100. Se estima que entre 0 y 10 es poco mantenible, entre 10 y 20 es moderadamente mantenible, y por encima de 20 es mantenible.

Para calcular este valor en nuestro proyecto se utilizó la herramienta *Ruby Critic* de *Ruby on Rails*.

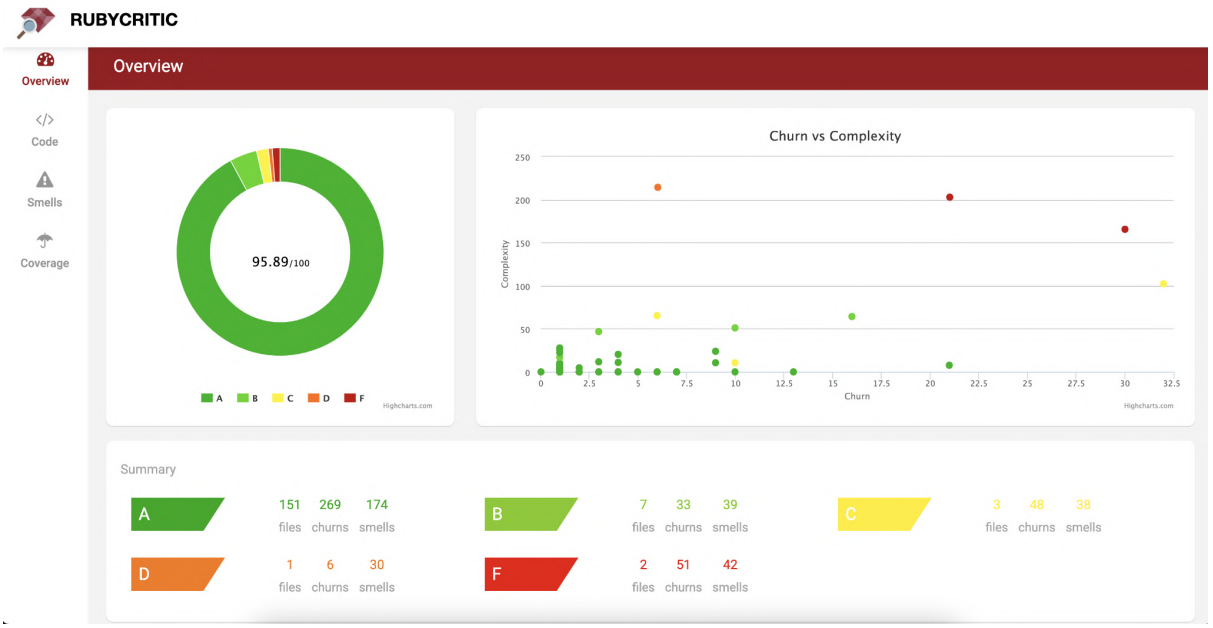


Figura 75: Índice de mantenibilidad del backend

Como se puede ver, el valor obtenido luego de ejecutar la herramienta es de 95.89 por lo cual se cumple con el objetivo planteado de tener un software mantenible utilizando la escala previamente mencionada.

Tiempo promedio para realizar tareas

Otra de las métricas utilizadas para el producto fue la de medir el tiempo promedio para realizar las distintas tareas. Es por esto que se realizaron pruebas al final de cada *release*, se pueden ver estas mismas en el Anexo 12.9. Pruebas de usabilidad (segundos y clicks).

Lo que se hizo fue pedirles a las mismas personas que realicen un flujo específico. Este flujo consiste en:

- Pedir una hamburguesa (scooby doo).
- Agregar gin tonic y una corona.
- Eliminar la corona.
- Finalizar pagando en efectivo.

Una vez que el usuario termina el pedido se registra la cantidad de tiempo que le llevó realizar esta tarea.

Se eligió realizarlo con las mismas personas para poder chequear el progreso de la usabilidad del sistema una vez que se implementan nuevos requerimientos y las mejoras sugeridas. Los resultados obtenidos fueron:

- En el primer *release* el tiempo promedio fue de 38 segundos.
- Segundo *release* logramos que baje a 35 segundos.
- El tercer *release* quedó en 30 segundos.

Algunos ejemplos de cambios que se realizaron para mejorar la usabilidad y aumentar la velocidad de los usuarios para realizar pedidos fueron:

- Implementar filtro por categorías:

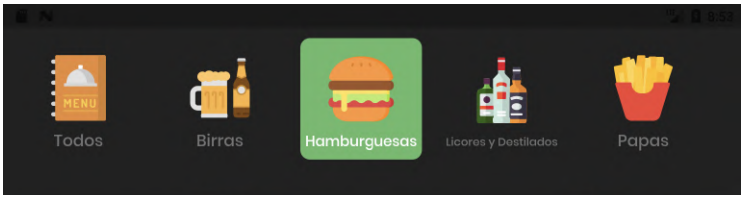


Figura 76: Categorías

- Implementar bottom modal sheet para pedir un producto en vez de que sea una pantalla nueva:

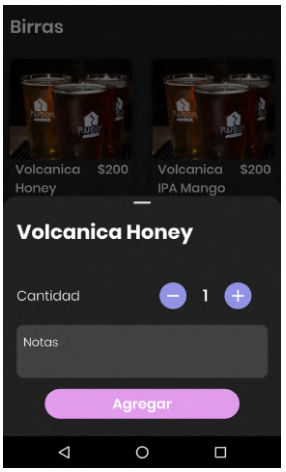


Figura 77: Bottom Modal Sheet

8.6. Piloto

Uno de los objetivos principales era la puesta en producción del proyecto, por lo que el equipo enfocó su trabajo para poder cumplir con esto.

Dicho esto, el sistema fue puesto en producción como piloto en el bar. El piloto consta de la parte administrativa y del tótem. Se colocaron dos tablets en el bar, una en cocina para recibir comandas, y la segunda es utilizada en una mesa a la cual suele ir gente conocida. Fue hecho de esta manera ya que es una prueba piloto y además que la tablet no se encuentra asegurada a la mesa.

Como uno de los objetivos del equipo era entregarle un sistema funcional que cumpla con todos los requisitos de calidad establecidos es que actualmente se encuentra en un piloto y no en producción. Este es el paso previo ya que el piloto se encuentra en un ambiente controlado siendo utilizado por usuarios conocidos del cliente.

De esta manera el equipo podrá recibir *feedback* de usuarios reales utilizando el sistema en un ambiente y flujo real. Si bien en las distintas etapas del proyecto se hicieron pruebas en el local, realizando un piloto se pueden encontrar nuevas cosas a tener en cuenta.

9. Gestión de la configuración

En el siguiente capítulo se presenta la configuración de software utilizada durante el proyecto. Se describen las herramientas utilizadas y las distintas decisiones para el gestionado de versiones y cambios.

9.1. Elementos de configuración

Se definen como elementos de configuración de *software* a todos los componentes de *software* o documentos que deben ser gestionados durante el transcurso del proyecto.

Se presentan a continuación los distintos elementos identificados con las herramientas asociadas.

Tipo	Elemento	Herramienta
Software	Código fuente *backend* y *frontend backoffice* (*Ruby on Rails*)	**GitHub**
	Código fuente aplicación móvil (*Flutter*)	
Documentación	Ingeniería de requerimientos	**Google Docs**
	Arquitectura	
	Gestión de proyecto	
	Plan de calidad	
	Plan de SQA	
	Gestión de riesgos	
	Gestión de configuración	
	Documentación final	**Word Office365**

Tabla 32: Elementos de configuración.

9.2. Herramientas para el control de versiones

9.2.1. Software

Se selecciona *Git* como el sistema de control de versiones del proyecto. Para tomar esta decisión el equipo hizo comparaciones con otras posibles herramientas.

Estás las pudimos clasificar en distribuidas o centralizadas, cada tipo tiene sus ventajas y desventajas. Finalmente creímos que lo más adecuado sería utilizar un sistema distribuido, el cual permite a cada integrante del equipo tener una copia local del repositorio, de esta manera se puede trabajar offline. Además de ser más utilizado en la comunidad por lo que se cuenta con más información.

Se utilizó *Github* como herramienta que facilita al equipo la colaboración con *Git*. Esta es una plataforma que puede mantener los repositorios en almacenamiento basado en la nube por lo que todos los integrantes pueden mantener el código actualizado. También el equipo ya contaba con experiencia previa en el uso de esta plataforma por razones de estudio y laborales.

9.2.2. Documentación

La gestión de documentos se hizo a través de la herramienta Google Docs. Esta permite el alojamiento y la edición compartida online de documentos similares a MS Word, MS Excel y MS PowerPoint. Es muy intuitiva, accesible y está en constante evolución. Además, permite con su funcionalidad de asignación de permisos es muy fácil compartir archivos con otras personas como el tutor y el cliente.

Previo a seleccionarla se probaron otras herramientas con similares características, pero está las supera en facilidad de uso y calidad de funcionalidades.

9.3 Repositorios

En esta sección se detalla cómo se organizaron los repositorios de código y documentación durante el proyecto.

9.3.1. Software

Teniendo en cuenta las distintas decisiones previas a comenzar con el desarrollo fue posible determinar qué repositorios eran necesarios para gestionar el código del sistema. Estos son los siguientes:

Repositorio	Descripción
tesis-app-back	Repositorio utilizado para almacenar el código del *backend* y *frontend* del *backoffice* en *Ruby on Rails*.
tesis-app-front	Repositorio utilizado para almacenar el código del *frontend* para la aplicación móvil, tótem, cocina y caja. Esto está en el mismo proyecto programado en Flutter.

Tabla 33: Repositorios

En la siguiente imagen se pueden ver ambos repositorios creados en GitHub.

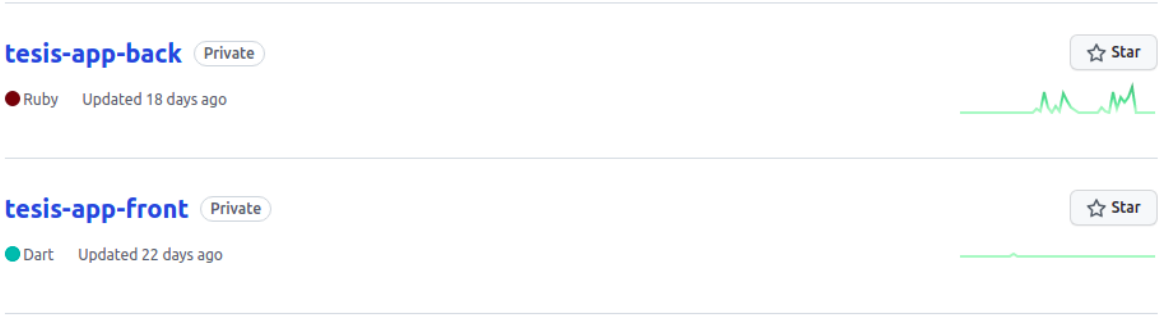


Figura 78: Repositorios, GitHub

9.3.2. Workflow

El equipo optó por utilizar el modelo de ramificación Gitflow como forma de trabajo con el repositorio. Implica el uso de ramas de funcionalidades y múltiples ramas primarias.

En este modelo los desarrolladores crean una rama *feature* donde implementan una nueva funcionalidad, luego cuando está terminada se hace merge de la rama *feature* con develop.

Una vez que develop cuenta con las suficientes funcionalidades para un *release*, se crea una nueva rama *release* partiendo de ese punto. No se puede agregar nuevas funcionalidades, solo arreglos de *bugs*. Cuando el *release* está pronto esta rama se debe hacer merge con main generando una nueva versión, también debe hacerse merge con la rama develop para actualizarla con los posibles arreglos hechos.

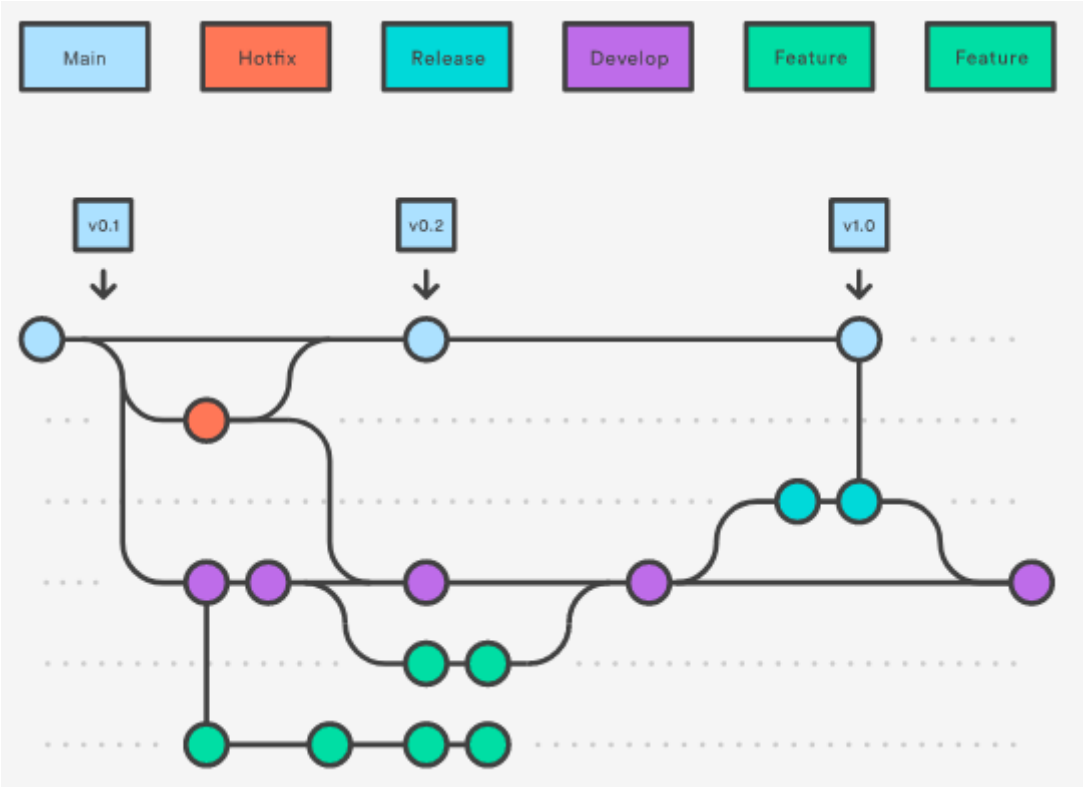


Figura 79: GitFlow [62]

Main

Rama que almacena todas las versiones de código testeadas y prontas para desplegar en producción.

Hotfix

Ramas de mantenimiento o “hotfix” que se usan para hacer arreglos rápidos sobre *releases* en producción.

Release

Rama que permite trabajar temporalmente sobre un nuevo lanzamiento de *release* en producción.

Develop

Rama a la cual los integrantes del equipo deben subir las funcionalidades que vayan desarrollando. Es importante que los desarrolladores prueben bien todo el sistema antes de subir un nuevo cambio para mantener la estabilidad de la rama develop.

Feature

Rama en la cual cada desarrollador trabaja para crear una nueva funcionalidad.

9.3.3. Documentación

Toda la documentación del proyecto fue guardada en carpetas en Google Drive compartidas entre los integrantes del equipo. A continuación, se muestra su estructura y que se puede encontrar en cada una de ellas.

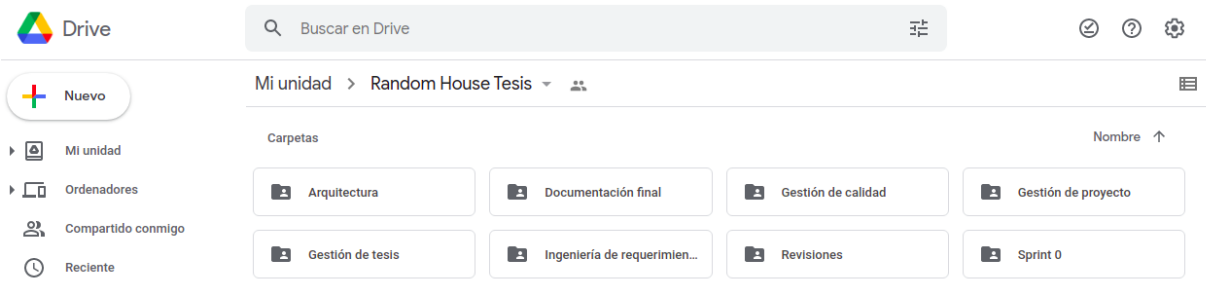


Figura 80: Gestión de documentación

Gestión de tesis

En esta carpeta se incluyen todos los documentos relacionados a las formalidades necesarias para iniciar la tesis. Formularios de presentación, escolaridades, curriculums, entre otros.

Sprint 0

Carpeta en la que se encuentran todos los documentos obtenidos de la etapa de investigación inicial. También de esta surgen otros documentos que se incluyen en sus categorías correspondientes. Aquí están los análisis de tecnologías, documentos que nos dio el cliente, entre otras cosas.

Arquitectura

Se incluyen todos los diagramas que se hacen en las etapas de diseño arquitectónico. Las distintas versiones que se fueron generando y sus documentos relacionados.

Ingeniería de requerimientos

En esta carpeta se encuentran todos los archivos obtenidos del análisis, especificación y validación de requerimientos. Estos son encuestas, entrevistas, conclusiones de *ingeniería inversa*. Se guarda aquí lo generado de las distintas validaciones con el cliente y las pruebas con los usuarios. También se incluyen *mockups* hechos a lo largo de los distintos ciclos.

Gestión de proyecto

Se guardan los documentos necesarios para llevar adelante la gestión de proyecto, estos son análisis de riesgos, metodologías de trabajo, métricas de gestión.

Gestión de calidad

Aquí se incluye el plan de calidad, plan de aseguramiento de calidad y plan de pruebas.

Revisiones

De las distintas revisiones que se realizaron durante el proyecto debimos construir presentaciones, borradores de textos de la presentación, guiones. Todo esto está en esta carpeta. Además, al finalizar cada revisión se generaron documentos con las mejoras propuestas por los revisores y los distintos informes que debíamos enviar.

Documentación final

Se realizaron varios documentos en forma de borradores para construir la entrega final del proyecto. Además de imágenes, links a bibliografía y más. Esto se puede encontrar en esta carpeta.

9.3.4. Gestión de cambios

Fue necesario establecer un proceso para validar nuevos cambios que vayan surgiendo en el correr del proyecto. Desde el comienzo hasta las últimas instancias del desarrollo nuevos cambios pueden surgir por parte del cliente o mismo de los integrantes del equipo.

Esto puede derivarse de distintas razones: iniciativa propia donde a alguien se le ocurre una idea de mejora, después de realizar pruebas de usabilidad (estas se pueden ver en Anexo 12.9. Pruebas de usabilidad, pruebas sobre *mockups*, mejoras detectadas por observación, entre otras.

Si estos cambios no son gestionados de la manera adecuada, pueden surgir problemas en la forma de trabajo del equipo. Es necesario que estos sean aprobados e incluidos de manera ordenada.

A continuación, se plantea el plan de gestión de un cambio.

1. Llega una solicitud de cambio
2. El *Product Owner* en conjunto con el resto del equipo deben evaluar si se debe hacer o no el cambio. Aquí se debe tener presente si el cambio aporta valor a la solución, si es adecuado según los objetivos del proyecto, si es viable su implementación según el impacto que genere.
3. Luego de evaluado, si se decide no seguir adelante con el cambio se descarta. Si fue propuesto por el cliente se le debe informar las razones de su descarte.
4. Por otro lado, si se decide incluir el cambio este se debe estimar en *story points* y priorizar dependiendo de su necesidad.
5. Por último, el cambio es agregado al *Product Backlog* para ser desarrollado en las siguientes iteraciones.

Figura 81: Gestión de cambios

9.4. Gestión de ambientes

Backend

Para el manejo de ambientes del *backend* se utilizó la herramienta Pipelines que provee Heroku. Un pipeline es un grupo de aplicaciones de Heroku que comparten el mismo código base. Cada una de estas representa una etapa del flujo de integración continua del desarrollo: review, staging, production.

Esto se utilizó de la siguiente manera, un integrante del equipo crea un pull request con una funcionalidad nueva, automáticamente Heroku crea una Review App permitiendo al equipo testear los nuevos cambios en una aplicación desechable.

Cuando estos cambios se consideran terminados van a ser integrados con la rama main siguiendo con el flujo mencionado la sección 9.3.2. Workflow. De aquí Heroku automáticamente despliega el código a la aplicación de staging para poder continuar con las pruebas necesarias. Luego cuando todo está pronto un integrante del equipo puede pasar la aplicación de staging a producción.

10. Conclusiones y lecciones aprendidas

En esta sección se encuentra la reflexión de lo que implicó la realización del proyecto junto con la explicación de hasta donde se llegó, como se pretende proceder y las lecciones aprendidas que le dejó al equipo.

10.1. Estado actual

El estado actual es el que se ha ido explicando en las secciones anteriores. De estas se desprende que nuestro cliente actualmente tiene como piloto dos tablets (una en cocina y una en una mesa) en donde los clientes (para el piloto solamente clientes seleccionados) pueden pedir a través de las mismas. En este piloto los clientes pueden pagar solamente con efectivo.

Además, se encuentra desplegado el panel administrativo para que puedan gestionar las comandas y la carta. La web de toma de comandas también se encuentra desplegada en caso de que se quiera extender la prueba haciendo uso de la misma.

Por último, se encuentra en la tienda de Android (Play Store) la aplicación disponible para ser descargada. La publicación en la tienda fue realizada para que el cliente la pueda probar y luego extender la prueba piloto a la aplicación mobile también.

A continuación, se presentan los resultados de los pedidos realizados a partir del 12/08/2021 fecha la cual se puso en marcha el piloto hasta el 12/09/2021.

En ese rango de fechas el bar se mantuvo abierto por 15 días en total, de los cuales el sistema se utilizó por grupo de conocidos del cliente 9 días.

En total se generaron un total de 38 pedidos.

Producto	Cantidad
Volcánica Blonde	14
Fernet	11

Random IPA	9
Random Blonde	8
Winnie the Pooh	5
Coca Cola	4
Margarita	4
Cabesas Bier Scottish	3
Oye Arnold	3
Pepperoni	2

Tabla 34: Productos más vendidos

Luego de este primer piloto, el equipo se puso en contacto con el cliente y con algunos usuarios que participaron del piloto. Pudimos concluir que el sistema se adaptó correctamente al flujo del bar y se encontraron algunas oportunidades de mejoras. Se consideró importante incluir otros métodos de pagos para facilitarle la compra aún más al cliente. Por otra parte, surgió como una inquietud de alguno de los usuarios, tener la posibilidad de acceder a un ticket de compra.

En cuanto al desarrollo del proyecto, se encuentra todo lo anterior terminado incluyendo también la integración para los pagos vía Mercado Pago en producción.

Con esto se concluye que fueron finalizadas todas las historias de usuario que se encontraban en el *backlog*. Por lo que el sistema cuenta con todos los requerimientos planteados al cliente.

10.2. Lecciones aprendidas

10.2.1. Gestión de proyecto

El haber llevado un seguimiento continuo del estado del proyecto a lo largo de toda su realización fue clave para poder detectar problemas. Las métricas de velocidad fueron esenciales para poder ajustar los desvíos de trabajo y mejorar en las siguientes iteraciones.

Haciéndole un seguimiento continuo a la distribución del esfuerzo hizo posible que cada integrante pueda ser consciente de su dedicación, mejorarla o notificarle a otro integrante en caso de existir algún desvío. Además, fue la única manera que se contaba para llevar un monitoreo de la distribución del tiempo en cada área de trabajo, esto es muy importante porque por ejemplo muchas veces nos enfocamos en el desarrollo y nos olvidamos de actualizar documentos u otras cosas.

El equipo pudo aprender que no hay que subestimar los riesgos ya que pueden afectar la velocidad y por lo tanto el desarrollo normal del proyecto. Cómo por ejemplo sucedió en período de evaluaciones.

10.2.2. Puesta en producción

Lo más importante a destacar es el trabajo que implica poner en producción de la mano del cliente un sistema. Si bien el sistema en teoría se encuentra terminado, ponerlo en producción lleva mucho trabajo, no solo que funcione todo tal y como estaba funcionando en desarrollo sino también todo lo referente a la capacitación e instalación en el local. Al mismo tiempo, hizo que el equipo se diera cuenta del cuidado que hay que tener cuando se tiene algo en producción ya que si por algún motivo alguno de los integrantes realiza una modificación que no estaba lo suficientemente probada, es un error que se traslada a producción y puede generar problemas para los usuarios que la están utilizando y gastos para el cliente.

10.2.3. Enfoque de trabajo

Creemos que utilizar una metodología ágil fue de gran ayuda para el desarrollo de nuestro proyecto ya que permitió llevar de cerca el trabajo que hacía cada integrante

del equipo y al mismo tiempo tener la capacidad de poder modificar o agregar requerimientos. Por la naturaleza del proyecto fue necesario adoptar esta metodología que permitía al equipo ser flexible en cuanto a los requerimientos y poder adaptarse fácilmente.

Conjuntamente con lo anterior es que utilizar las herramientas correctas resulta ser muy importante y afecta directamente en la calidad del producto y en la productividad del equipo. Durante el transcurso de este proyecto el equipo debió cambiar la herramienta que utilizaba en el día a día como herramienta de gestión y seguimiento del proyecto y *Scrum*. Esto se debió a que el equipo encontró una herramienta que era más completa para las necesidades requeridas, por lo que se decidió utilizar *Jira* como sustituto. Esta última resultó ser una herramienta mucho más completa la cual ofrecía *features* muy útiles para el seguimiento y control de *Scrum*. También facilitó a la hora de realizar seguimiento de las métricas del equipo. Además, estas herramientas permitieron al equipo tener el backlog ordenado lo cual permitió al equipo trabajar de una forma más prolija.

En cuanto al ciclo de vida nos pareció acertado la utilización del iterativo incremental ya que le permitió al equipo entregar un prototipo funcional en corto tiempo y al cliente tener la posibilidad de probar y entregar *feedback* lo cual resultó muy conveniente para los siguientes *releases*.

10.2.4. Ingeniería de requerimientos

El equipo logró ver la importancia de la validación (se pudo descartar funcionalidades que no iban a aportar tanto al modelo de negocio como por ejemplo los juegos) ya que se enfocó en las funcionalidades que realmente mejoraron el modelo de negocio y no en otras que el cliente en principio pensaba que aportarían, pero que debido al alcance no terminó siendo prioritario.

En cuanto a las pruebas con usuarios logramos reafirmar que son cruciales para el desarrollo de un sistema. Gracias a esto se logró concluir en un producto amigable y usable en el contexto del bar.

Tener un cliente inexperto técnicamente complejiza a la hora de que el equipo explique o quiera justificar algún cambio o requerimiento nuevo, lo que implicó un trabajo extra que el equipo no tenía previsto en un principio.

Otra lección que hubo fue que es igualmente de complejo como importante lograr empatizar con los distintos usuarios. En este caso se tenía cuatro tipos de usuarios con distintos contextos (por contextos se entiende como: condición lumínica, dispositivo que utilizará, tiempo que dispone, etc.) en donde cada uno tenía que tener un flujo optimizado para su uso, ya sea tanto los cocineros para no perder tiempo en el control de las comandas, como los usuarios a la hora de realizar los pedidos, como los administradores a la hora de controlar la carta o de controlar promociones o cupones, o como los cajeros a la hora de tomar comandas.

10.2.5. Arquitectura y tecnologías

Como el cliente no era experto en lo técnico, la elección de las tecnologías quedó 100% a cargo del equipo. Esto llevó a que se tenga que realizar una investigación en donde se compararon las distintas alternativas, en general el equipo estaba acostumbrado a utilizar tecnologías que se les fueran dadas. Por lo cual habiendo hecho esta investigación el equipo logró darse cuenta de la importancia de cómo cada tecnología favorece ciertas áreas, esto hace que sea muy importante elegir correctamente para el dominio en el cual se trabaja.

Otra de las lecciones fue la importancia de entender correctamente el dominio para luego poder pensar en la arquitectura, si uno no termina de entender el problema es probable que se tenga que cambiar la arquitectura luego de comenzado el desarrollo del producto, lo cual es muy costoso.

Es importante también tomar en cuenta la curva de aprendizaje de cada tecnología ya que no todos los integrantes de un equipo tienen porque saber esa tecnología, y ese tiempo de aprendizaje hay que saber gestionarlo ya que no se avanza en el desarrollo de funcionalidades.

10.2.6. Gestión de calidad

Esta era un área en la cual por primera vez el equipo se iba a ver responsable 100% lo cual implicó un desafío ya que ninguno tenía mucha experiencia en el tema. Fue clave tener desde un principio bien definido cómo se iba a gestionar la calidad para así poder ir controlando en cada iteración.

Sin lugar a dudas haber seguido rigurosamente el plan y las actividades nos hicieron llegar a un producto en el cual creemos que cumple con todos los requisitos de calidad y que además el cliente quedó satisfecho.

Llegamos a la conclusión de que, en proyectos tan largos como estos, si no se es cuidadoso y no se presta atención a la calidad durante el proyecto por más que en el resto de las áreas se haya realizado un buen trabajo es difícil llegar al resultado esperado.

10.3. Conclusiones

10.3.1. Conclusiones sobre los objetivos académicos

- Poner en práctica todo lo aprendido en la carrera

  Todo lo aprendido en la carrera fue aplicado en este proyecto en sus distintas áreas, permitiendo al equipo llevar a cabo las diversas tareas con éxito. Consideramos que fue un proyecto en donde se pudo aplicar todo lo aprendido en un solo lugar lo cual resultó muy provechoso.

- Aprender nuevas tecnologías

  Dicho objetivo fue importante para que el equipo tenga un desafío extra y pueda aprender tecnologías que previamente no manejaban, lo cual es importante ya que son nuevas tecnologías que se agrega al conocimiento de cada integrante las cuales pueden ser aplicadas a futuro en el ámbito laboral. El equipo cumplió con el objetivo propuesto sobre la cantidad de tecnologías a aprender, ya que se había planteado aprender al menos dos y se terminaron aprendiendo las siguientes: *Flutter*, *Ruby on Rails, Firebase, Heroku.*

10.3.2. Conclusiones sobre los objetivos del equipo

- Aprender a trabajar en equipo aplicando metodologías de trabajo

  Este objetivo era muy importante para el equipo ya que se trataba de aplicar lo aprendido en la carrera en un equipo creado y gestionado por nosotros. Lo que implica mucha responsabilidad y lograr ser metódicos a la hora de realizar las tareas. Para esto fue necesario asignar roles a los integrantes del equipo en donde cada integrante debía controlar la correcta realización de distintas tareas. Creemos que el equipo logró trabajar de forma satisfactoria, esto fue medido a través de una encuesta y los resultados se pueden ver en el Anexo 12.10. Encuesta de satisfacción del trabajo en equipo.

- Aprender a tratar con clientes reales

  Para el equipo este objetivo era completamente nuevo ya que en las experiencias previas tanto laborales como en la universidad no se tuvo que tratar de primera mano con clientes reales y tomar las decisiones acordes para lograr la correcta realización del producto.

10.3.3. Conclusiones sobre los objetivos del producto

- Crear un producto de valor para el cliente generando impacto en su negocio

  En base a la entrevista de satisfacción que se realizó al cliente creemos que se pudo lograr el objetivo planteado.

- Ser innovadores en el segmento del mercado

  El equipo cree en base a lo investigado frente a la competencia se pudo lograr un sistema integral en el cual todos los pedidos pasan por el mismo y no solo hace que se diferencie el sistema de parte de los usuarios sino también el sistema de gestión para la cocina y para los administradores.

- Crear un sistema intuitivo para los clientes del bar

  En base a las pruebas realizadas con usuarios y al *feedback* recibido, el equipo pudo concluir que se pudo lograr con éxito este objetivo lo que afecta

directamente en la satisfacción de los clientes.

10.3.4. Conclusiones personales

Rodrigo Duarte

“En cuanto al desarrollo de la tesis considero que fue un proyecto muy importante para mi experiencia tanto educativa como profesional. En muchos años que tengo de trabajo como desarrollador de software nunca antes había hecho un proceso completo desde la programación hasta la liberación del sistema, como también toda su gestión.

Además, nunca antes había trabajado antes con una metodología ágil y con sus herramientas por lo que desde este punto me aportó mucho, aprendí cómo es su funcionamiento y me considero capaz de aplicarlo en el mundo laboral.

Sumado a lo anterior, me fue muy relevante experimentar lo que es el relacionamiento con un cliente. El proceso de recabar y analizar los requerimientos en conjunto con un cliente es algo que tampoco había hecho antes y me sirvió mucho para aprender cosas que nunca se pueden predecir hasta vivirlas en la práctica. El trato no siempre es tan sencillo como parece y todo lo obtenido de esta experiencia puede servir mucho para el futuro donde me tenga que enfrentar a situaciones similares.

En conclusión, creo que la experiencia de realizar el proyecto de grado es muy importante para terminar de afianzar los conocimientos aprendidos durante toda la carrera, ver los resultados positivos es algo muy gratificante considerando el arduo trabajo que se le dedicó.”

Juan Drets

“Creo que la tesis nos sirvió para poder integrar y cerrar todo lo aprendido a lo largo de la carrera. Es una buena manera de poder integrar lo teórico y práctico con la vida real, ya que pudimos aplicar muchas de las cosas que aprendimos en un proyecto para un cliente real. Eso implicó una responsabilidad extra ya que el resultado afectaría al cliente directamente. Por suerte con el equipo, a pesar de ser

la primera vez que trabajamos todos juntos, pudimos entregar un proyecto sólido y completo que logró satisfacer a nuestro cliente.

De un proyecto de estas características puedo resaltar algunos aprendizajes o conclusiones como puede ser que tratar con clientes reales dista mucho de un proyecto universitario, me sirvió para entender que las habilidades blandas y el poder comprender realmente al cliente es igual de importante que si se es un buen desarrollador. Saber elegir las tecnologías correctas también es muy importante ya que implica entender al máximo al cliente y sus proyecciones para con el proyecto, ya que un proyecto de este tipo puede conllevar una relación larga con el cliente.

En síntesis, creo que tener una experiencia como esta es super enriquecedora y sirve para darle un cierre a la carrera pudiendo conocer todos los aspectos que la involucran.”

Sebastian Rodriguez

“Personalmente creo que este proyecto fue la mejor forma que tuve para ir poniéndole un punto final a esta etapa de estudiante de Ingeniería en Sistemas en la Universidad ORT.

A lo largo de más de un año que fue lo que este proyecto nos llevó, fuimos aplicando todo lo aprendido en la carrera en un proyecto de mayor magnitud que un obligatorio como estábamos acostumbrados, y lo que no es menos importante, a un cliente real con todo lo que esto implica. Durante todo este año, con el equipo nos íbamos dando cuenta cómo las distintas cosas que vimos en las diferentes materias a lo largo de la carrera se iban aplicando en el mundo real.

Por otro lado, también destacar que era la primera vez realizando un trabajo con los otros miembros del equipo, y a pesar de esto, desde el primer momento pudimos trabajar como un equipo dividiendo tareas y estando todos sincronizados. Así fue como pudimos llegar a un resultado final con el cual estamos más que contentos.

En conclusión, creo que el proyecto de grado es la mejor forma en la cual un estudiante puede finalizar su carrera. Va a ser lo más cercano a un trabajo fuera de la facultad, en el cual uno se da cuenta de todas las responsabilidades que esto

implica y al mismo tiempo hace que te des cuenta de la importancia de trabajar en equipo para llegar a buenos resultados.”

10.4. Próximos pasos

Como próximos pasos el equipo tiene planteado poner en producción las partes del sistema que no se encuentran dentro del piloto, estos son: la aplicación móvil y la web de toma de pedidos en la caja. Si bien esta web se encuentra desplegada, no ha sido usada en producción. En cuanto a la aplicación móvil resta publicar en la tienda para iOS (App Store). Además, resta habilitar el pago vía Mercado Pago en la terminal autoservicio.

Resta evaluar en conjunto con el cliente si es de valor incorporar la opción de imprimir tickets del pedido para los clientes. Además de esto en un futuro habría que integrarse con algún sistema de facturación electrónica.

Por último, resta definir el modelo de negocio que tendrá el sistema y la relación que mantendrá el equipo con Random House.

11. Referencias bibliográficas

[1] TIC Portal, “Add-on: ¿Qué es y cómo hace para ampliar las funcionalidades?,” Feb-2021, [Online]. Available: https://www.ticportal.es/glosario-tic/add-on. [Accessed: Nov-2021].

[2] RedHat, “¿Qué es una api?” [Online]. Available: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Accessed: Nov-2021].

[3] Platzi, “Qué es frontend y backend: Diferencias y ... - platzi.” [Online]. Available: https://platzi.com/blog/que-es-frontend-y-backend/. [Accessed: Nov-2021].

[4] Ecured, “Beeper”, [Online]. Available: https://www.ecured.cu/Beeper. [Accessed: Nov-2021].

[5] Techopedia, “What is software bug? - definition from Techopedia”, Jan-2017, [Online]. Available: https://www.techopedia.com/definition/24864/software-bug-. [Accessed: Nov-2021].

[6] Agile Nutshell, “Burndown charts”, [Online]. Available: http://www.agilenutshell.com/burndown. [Accessed: Nov-2021].

[7] Smartbear, “What is code review?”, [Online]. Available: https://smartbear.com/learn/code-review/what-is-code-review/. [Accessed: Nov-2021].

[8] Atlassian, “Scrum: Qué es, cómo funciona y por qué es excelente”, [Online]. Available: https://www.atlassian.com/es/agile/scrum. [Accessed: Nov-2021].

[9] Cyberclick, “¿Qué es un dashboard y para qué se USA? (2022)”, [Online]. Available: https://www.cyberclick.es/numerical-blog/que-es-un-dashboard. [Accessed: Nov-2021].

[10] CS, “Vol. 8, no. 4, July{August 2009 an overview of feature ...”, [Online]. Available: https://www.cs.cmu.edu/~ckaestne/pdf/JOT09_OverviewFOSD.pdf. [Accessed: Nov-2021].

[11] Rock Content, “Feedback: ¿qué es y cuál es su importancia en Las Empresas?”, Feb-2021, [Online]. Available: https://rockcontent.com/es/blog/que-es-feedback/. [Accessed: Nov-2021].

[12] Firebase Google, “Firebase te ayuda a compilar y ejecutar apps exitosas”, [Online]. Available: https://firebase.google.com/?hl=es. [Accessed: Nov-2021].

[13] Flutter, “Beautiful native apps in record time”, [Online]. Available: https://flutter.dev/. [Accessed: Nov-2021].

[14] Wikipedia, “Front-end web development”, Nov-2021, [Online]. Available: https://en.wikipedia.org/wiki/Front-end_web_development. [Accessed: Nov-2021].

[15] Arimetrics, “Qué es framework - Definición, Significado y Ejemplos”, Aug-2021, [Online]. Available: https://www.arimetrics.com/glosario-digital/framework. [Accessed: Nov-2021].

[16] Le Wagon, “Full stack developer job profile - what does a full stack developer do?: Le wagon”, [Online]. Available: https://www.lewagon.com/tech-jobs/web-development/full-stack-developer. [Accessed: Nov-2021].

[17] Git, “Git is a free and open source distributed version control system”, [Online]. Available: https://git-scm.com/. [Accessed: Nov-2021].

[18] Google, “Proteger sitios con el protocolo HTTPS”, [Online]. Available: https://developers.google.com/search/docs/advanced/security/https?hl=es. [Accessed: Nov-2021].

[19] Ingenieros Asesores, “Ingeniería Inversa: Concepto Y Aplicaciones Ingenieros Asesores”, Aug-2020, [Online]. Available: https://ingenierosasesores.com/actualidad/ingenieria-inversa-concepto-aplicaciones/. [Accessed: Nov-2021].

[20] IBM, “Formato JSON (JavaScript object notation)”, [Online]. Available: https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format. [Accessed: Nov-2021].

[21] Atlassian, “Jira: Software de seguimiento de proyectos E incidencias”, [Online]. Available: https://www.atlassian.com/es/software/jira. [Accessed: Nov-2021].

[22] Material Design, “Design”, [Online]. Available: https://material.io/design. [Accessed: Nov-2021].

[23] Imborrable, “Mockup: Qué es y para qué se usa en diseño gráfico”, Jun-2021. [Online]. Available: https://imborrable.com/blog/mockup-que-es/. [Accessed: Nov-2021].

[24] InboundCycle, “¿Qué es el MVP o producto mínimo viable?”, [Online]. Available: https://www.inboundcycle.com/blog-de-inbound-marketing/bid/189238/qu-es-el-mvp-o-producto-m-nimo-viable. [Accessed: Nov-2021].

[25] Javier Garzas, “La técnica del planning poker”, Jan-2018, [Online]. Available: https://www.javiergarzas.com/2018/01/la-tecnica-del-planning-poker.html. [Accessed: Nov-2021].

[26] Scrum, “What is a product owner?”, [Online]. Available: https://www.scrum.org/resources/what-is-a-product-owner. [Accessed: Nov-2021].

[27] Programación y más, “Scrum: ¿Qué es el product backlog?”, [Online]. Available: https://programacionymas.com/blog/scrum-product-backlog. [Accessed: Nov-2021].

[28] Medium, “¿Qué son los pull requests y por qué valen la pena?”, [Online]. Available: https://blog.nearsoftjobs.com/qu%C3%A9-son-los-pull-requests-y-por-qu%C3%A9-valen-la-pena-ed98d19d124e. [Accessed: Nov-2021].

[29] VWO, “What are push notifications? A complete guide [up-to-date]”, [Online]. Available: https://vwo.com/push-notifications/#:~:text=Push%20notifications%20are%20clickable%20pop,other%20information%20to%20their%20customers. [Accessed: Nov-2021].

[30] Lenguaje de Programación Ruby, “Ruby”, [Online]. Available: https://www.ruby-lang.org/es/. [Accessed: Nov-2021].

[31] Guides, “Rubygems guides”, [Online]. Available: https://guides.rubygems.org/. [Accessed: Nov-2021].

[32] Ruby on Rails, “Ruby on rails”, [Online]. Available: https://rubyonrails.org/. [Accessed: Nov-2021].

[33] Plutora, “What is a software release? (all that you need to know)”, [Online]. Available: https://www.plutora.com/software-release-management/software-release. [Accessed: Nov-2021].

[34] Red Hat - We make open source technologies for the enterprise, “¿Qué es una api de rest?,” [Online]. Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Accessed: Nov-2021].

[35] Wikipedia, “Gestión de Configuración de software”, [Online]. Available: https://es.wikipedia.org/wiki/Gesti%C3%B3n_de_configuraci%C3%B3n_de_software. [Accessed: Nov-2021].

[36] Ministerio de Salud Pública, “Covid-19 en uruguay del 13 de Marzo al 13 de setiembre”, [Online]. Available: https://www.gub.uy/ministerio-salud-publica/comunicacion/noticias/covid-19-uruguay-del-13-marzo-13-setiembre. [Accessed: Nov-2021].

[37] WhatsApp, “WhatsApp features,” [Online]. Available: https://www.whatsapp.com/features/. [Accessed: Nov-2021].

[38] Delysoft, “Administre su comercio gastronómico con Delysoft.”, http://www.delysoft.com/. [Accessed: Nov-2021].

[39] Firebase Google, “Firebase authentication | firebase documentation”, [Online]. Available: https://firebase.google.com/docs/auth. [Accessed: Nov-2021].

[40] Firebase Google, “Cloud storage for firebase | firebase documentation”, [Online]. Available: https://firebase.google.com/docs/storage. [Accessed: Nov-2021].

[41] Mercado Pago, “Mercado Pago: Hacemos Que Cobrar y pagar no tenga límites”, [Online]. Available: https://www.mercadopago.com.uy/. [Accessed: Nov-2021].

[42] Chartkick, “Create beautiful javascript charts with one line of ruby”, [Online]. Available: https://chartkick.com/. [Accessed: Nov-2021].

[43] Heroku, “What is Heroku”, [Online]. Available: https://www.heroku.com/what. [Accessed: Nov-2021].

[44] Bugsnag, “Web app monitoring software”, [Online]. Available: https://www.bugsnag.com/solutions/web. [Accessed: Nov-2021].

[45] Medium, T. M. D., “What is UI vs. UX Design? what's the difference?”, [Online]. Available: https://uxplanet.org/what-is-ui-vs-ux-design-and-the-difference-d9113f6612de. [Accessed: Nov-2021].

[46] Google play, “Google play”, [Online]. Available: https://play.google.com/store. [Accessed: Nov-2021].

[47] Software Testing Help, “What is software quality assurance (SQA): A guide for beginners”, Nov-2021. [Online]. Available: https://www.softwaretestinghelp.com/software-quality-assurance/. [Accessed: Nov-2021].

[48] Menoo, “Menoo en la cocina y en tu celular”, [Online]. Available: https://www.menoo.com.uy/. [Accessed: Nov-2021].

[49] Soft Restaurant® Uruguay, “Inicio”, [Online]. Available: http://www.softrestaurant.com.uy/. [Accessed: Nov-2021].

[50] PedidosYa, “Todo lo que necesites, ¡Te Lo llevamos!”, [Online]. Available: https://www.pedidosya.com.uy/. [Accessed: Nov-2021].

[51] Reactive Programming, “Arquitectura monolítica”, [Online]. Available: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/monolitico. [Accessed: Nov-2021].

[52] Brocoders blog about software development, “Flutter vs. react native in 2021 - detailed framework comparison”, [Online]. Available: https://brocoders.com/blog/flutter-vs-react-native/. [Accessed: Nov-2021].

[53] Nielsen Norman Group, “10 usability heuristics for user interface design”, [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed: Nov-2021].

[54] Wave, “Wave Web Accessibility Evaluation Tool”, [Online]. Available: https://wave.webaim.org/. [Accessed: Nov-2021].

[55] Swagger, “Bearer authentication”, [Online]. Available: https://swagger.io/docs/specification/authentication/bearer-authentication/. [Accessed: Nov-2021].

[56] JSON Web Tokens, “Jwt.io”, [Online]. Available: https://jwt.io/. [Accessed: Nov-2021].

[57] Amazon, “Compute”, [Online]. Available: https://aws.amazon.com/compute/sla/. [Accessed: Nov-2021].

[58] Heroku Elements, “Elements marketplace: Heroku-load-balancer”. [Online]. Available: https://elements.heroku.com/buttons/smartpension/heroku-load-balancer. [Accessed: Nov-2021].

[59] R. C. Martin, *Clean code*. Massachusetts: Pearson Education, Inc., 2008.

[60] Medium, “Flutter: Best practices and tips”, May-2021, [Online]. Available: https://medium.com/flutter-community/flutter-best-practices-and-tips-7c2782c9ebb5. [Accessed: Nov-2021].

[61] RailsCarma, “Rails development: Coding Conventions & Best Practices - Railscarma - Ruby on rails development company specializing in offshore development”, Jun-2021, [Online]. Available: https://www.railscarma.com/blog/technical-articles/rails-development-coding-conventions-best-practices/. [Accessed: Nov-2021].

[62] Atlassian, “Gitflow workflow: Atlassian Git Tutorial”, [Online]. Available: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. [Accessed: Nov-2021].

12. Anexos

12.1. Informes de revisiones

12.1.1. Informe de revisión 1

Grupo:	Tótems de órdenes para bar		**Fecha:**	02/12/20
Carrera:	Ing. Sistemas		**Revisores:**	Leonardo Scafarelli

Fortalezas del grupo

El sistema de pedidos que actualmente implementa el cliente es muy engorroso por lo cual esto lo vemos como una oportunidad para mejorar la experiencia de los clientes y que impacte positivamente en el bar.

Oportunidades de mejora

- Validación con potenciales usuarios del tótem.
- Mejorar el uso de las metodologías.
- Reevaluar la parte de design thinking.
- Definir *sprints*.
- Detallar cronograma con actividades realizadas y actividades a realizar.
- Contactarse con grupo de proyecto final que realizó un trabajo similar al nuestro

Acciones a realizar

#	**Descripción de la Acción**	**Fecha estimada**
1	Definir si se va a seguir utilizando design thinking	16/12/2020
2	Definir bien cómo vamos a implementar las metodologías	16/12/2020
3	Validar el tótem con potenciales usuarios	28/12/2020
4	Definir sprints	16/12/2020
5	Realizar un cronograma de actividades	16/12/2020

6	Contactarse con el otro grupo de proyecto final	26/12/2020

12.1.2. Informe de revisión 2

Grupo:	Random House		**Fecha:**	19/04/21
Carrera:	Ing. Sistemas		**Revisores:**	Rafael Bentancur

Fortalezas del grupo

Sobre la presentación las ppts fueron completas y fáciles de entender. Presentamos una demo representativa del sistema y también fácil de entender. Las metodologías y técnicas fueron bien aplicadas.

Oportunidades de mejora

- Cerrar el ciclo entre los atributos de calidad y el plan de calidad junto con el plan de aseguramiento de calidad.
- Agregar las métricas de usabilidad y disponibilidad.
- Realizar pruebas de estrés.
- Analizar no solo la performance del *frontend* sino que también del *backend*.
- En vez de promediar la velocidad, calcular el mínimo y máximo.
- Agregar diapositiva de lecciones aprendidas y conclusiones.

Acciones a realizar

#	Descripción de la Acción	Fecha estimada
1	Analizar atributos de calidad y mejorar plan de calidad	25/04/2021
2	Presentar métricas de usabilidad	25/04/2021
3	Agregar diapositiva de lecciones aprendidas y conclusiones.	Próxima revisión

4	Calcular minimo y maximo de las velocidades	03/05/2021
5	Realizar pruebas de estrés	02/05/2021
6	Analizar la performance del back	02/05/2021

12.1.3. Informe de revisión 3

Grupo:	Random House		**Fecha:**	24/8/21
Carrera:	Ing. Sistemas		**Revisores:**	Amalia Álvarez

Fortalezas del Grupo

Sobre la presentación la misma fue clara y fácil de entender. Presentamos una demo representativa del sistema que ayudó a comprender la magnitud y dificultad de nuestra solución. Por otro lado, se destacó la cantidad de métricas que se utilizaron.

Oportunidades de Mejora

- Discutir la denominación del ciclo de vida.
- Mejorar el esquema del *Customer Journey*
- Hacer más énfasis en la puesta a producción del sistema.
- Explicar mejor la selección de tecnologías.
- Mejorar los diagramas de arquitecturas.
- Explicar el tema de quién es el dueño del producto.
- Mejorar comunicación en general.
- Agregar requerimientos funcionales y no funcionales en la presentación.

Acciones a Realizar

#	Descripción de la Acción	Fecha estimada

1	Discutir la denominación del ciclo de vida.	28/08/2021
2	Mejorar el esquema del *Customer Journey*	28/08/2021
3	Hacer más énfasis en la puesta a producción del sistema.	Defensa
4	Explicar mejor la selección de tecnologías.	05/09/2021
5	Mejorar los diagramas de arquitectura.	05/09/2021
6	Explicar el tema de quién es el dueño del producto.	Defensa
7	Mejorar comunicación en general.	Defensa
8	Agregar requerimientos funcionales y no funcionales en la presentación.	Defensa

12.2. Informe de avance

ORTsf	**Informe de Avance del Proyecto**	

Random House

Objetivo

Informar sobre el estado del proyecto a través los datos sobre la situación actual del proyecto, los resultados de las actividades realizadas hasta el momento y el pronóstico hasta el final del proyecto.

Alcance

Este documento es aplicable al proyecto *Random House.*

Se basa en información recopilada desde el comienzo del proyecto hasta el *15-04-2021*, fecha a la que se hace referencia en cada tabla de información, como fecha de corte del proyecto para informar su estado.

Agosto 2020

Revisado: *[nombre]*	*[fecha]*
Aprobado: *[nombre]*	*[fecha]*
Archivo:	

1. Descripción del proyecto

Hoy en día se puede llegar a perder mucho tiempo a la hora de realizar un pedido en bares, mercados o locales de comida rápida.

El sistema de ventas no cuenta con atención a la mesa, sino que los clientes deben realizar el pedido, esperarlo y luego retirarlo en la barra. Esto genera que muchas veces se cuenta con una sola persona atendiendo a los clientes y puede llegar a generar largas filas de espera repercutiendo negativamente en la experiencia que significa salir a tomar y comer. De todas formas, debido a la situación sanitaria actual, por el momento se está contando con mozos en el bar los cuales entregan los pedidos a la mesa, pero apenas se normalice todo se volverá tener que retirar los pedidos por la barra.

El proyecto busca resolver este problema y más, mediante la implementación de tótems en los locales, que servirán para que los clientes puedan acercarse a ellos y realizar sus pedidos. Se pueden instalar varios en simultáneo para así poder descongestionar las largas filas que se forman cuando se tiene a una persona atendiendo.

Por otra parte, se le agregara valor a los clientes mediante el desarrollo de una aplicación móvil que busca generar fidelidad con los clientes, mejorando su experiencia en el bar. En ella se incluyen la posibilidad de realizar pedidos, dividir la cuenta, recibir promociones y cupones de descuentos, entre otras funcionalidades.

El flujo de la terminal de autoservicio sería el siguiente:

Flujo terminal autoservicio:

El cliente solicita su pedido en el totem, esté confirma luego de realizar el pago y se le solicita que se le asigne un beeper (dispositivo electrónico que suena y vibra al estar pronta la orden), al sonar el cliente se entera de que su orden está pronta y puede ir a retirarla a la barra. Retorna el beeper y concluye el flujo de atención. Al mismo tiempo se envía una comanda a la cocina para ser visualizada y tener la lista de pedidos a disposición. A su vez, se expondrá un endpoint para permitir la

integración con nuestro sistema y de esta manera obtener en tiempo real los pedidos.

Flujo compra por aplicación:

Por medio de la aplicación el usuario tiene la opción de realizar varios pedidos en su estadía en el bar y luego al final realizar el pago de la totalidad.

El cliente solicita su pedido por medio de la aplicación, ingresa el número de mesa en el que se encuentra y decide si hacer varios pedidos y luego pagar, o pagar directamente. El nuevo pedido es enviado como una comanda a la cocina.

Aquí se desprenden dos situaciones:

1. En modalidad por la pandemia, cuando el pedido está pronto un mozo se lo lleva a la mesa del cliente
2. En modalidad normal, se le notifica al cliente que el pedido se encuentra pronto y lo retira en la barra.

Otra parte importante del sistema es un backoffice en donde el personal del bar puede agregar, editar y eliminar usuarios, categorías de productos, productos, comandas, promociones y cupones. Como también visualizar un *dashboard* y descargar reportes con analítica de los consumos, generando información de valor al bar. Esto incluye cuando los usuarios consumen más, que tipos de productos se consumen más que otros y así permitir al bar tomar decisiones sobre el negocio.

2. Carrera e integrantes del equipo

Carrera: Ingeniería en sistemas
Integrantes: Juan Drets (209326), Sebastián Rodríguez (192412), Rodrigo Duarte (217374)

3. Situación actual del proyecto

El proyecto fue aprobado a finales de septiembre del 2020 por lo que se comenzó con la etapa cero (*sprint* 0) del proyecto en octubre de ese mismo año. Este incluye investigación y planificación previo a la primera revisión y comienzo del desarrollo.

Dado que se definió un proceso evolutivo con un enfoque de prototipado, se desarrolla rápidamente un primer *MVP* con la finalidad de definir los requerimientos que aún no estaban claros para el cliente, el cual se concluye en el primer *release*.

Ya que la dimensión del trabajo a realizar va a depender no únicamente de lo ya relevado, sino también de lo obtenido luego de cada ciclo evolutivo, se estima que se presenta un prototipo al cliente en aproximadamente 4 o 6 *sprints* de desarrollo.

En total se planificaron tres *releases* para desarrollar la totalidad de la aplicación de usuario y terminal de autoservicio.

Actualmente el proyecto cuenta con los siguientes requerimientos funcionando:

- Login en aplicación
- Registro de usuario
- Alta, baja y modificación de productos
- Alta, baja y modificación de usuarios (en la parte administrativa)
- Alta, baja y modificación de categorías
- Alta, baja y modificación de promociones
- Alta, baja y modificación de promociones destacadas
- Alta, baja y modificación de cupones
- Flujo para realizar un pedido (buscar los productos en el menú, pudiendo filtrar por categorías. seleccionar los productos que se deseen pudiendo ver la descripción de los mismos y agregarlos al carrito)
- Pantalla de promociones
- Pantalla de menú
- Pantalla de carrito. Además de visualizar los elementos que se han agregado se permite que el usuario pueda sumar o restar más productos de los que agregó y también que los pueda quitar del carrito.
- Pantalla de selección de beeper (totem)
- Pantalla de selección de mesa (app mobile)
- Alta, baja y modificación de un nuevo negocio
- Alta, baja y modificación de comandas
- Pantalla de comandas. Visualización y acciones sobre comandas desde la aplicación en la cocina.

La última funcionalidad mencionada aún no se encuentra finalizada. Cuenta con la lógica en el *backend* y la comunicación con la aplicación mobile, pero el desarrollo de la pantalla en la aplicación mobile se encuentra desarrollada con un funcionamiento básico. Esto se mejorará ya que es una pantalla que requiere varias iteraciones con los usuarios, cocineros en este caso, para que sea usable y práctica en el día a día.

Durante el desarrollo de las funcionalidades, en paralelo, se realizaron tareas de gestión fijas. Incluyendo reuniones diarias (algunas presenciales y otras no), al final de cada *Sprint*, análisis de riesgos, métricas y pruebas funcionales.

Para explicar lo realizado más a detalle, se presenta una tabla donde se pueden ver las funcionalidades y tareas realizadas especificando el esfuerzo que llevaron realizarlas.

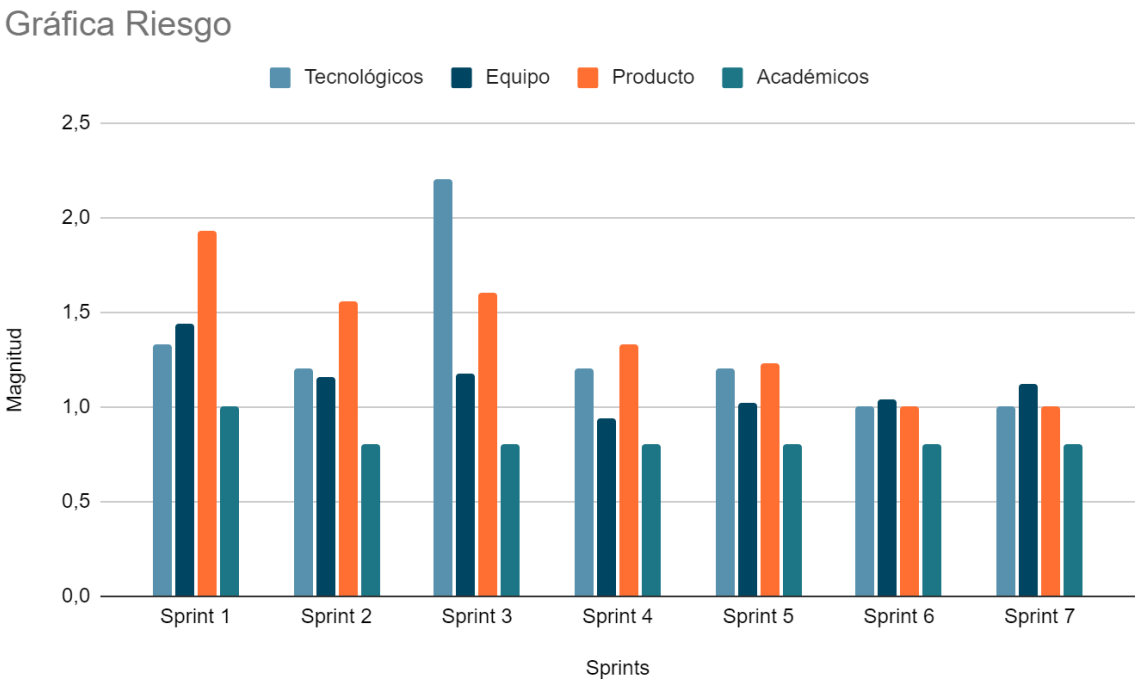
En esta se puede ver un *sprint* 0, donde se realizaron tareas de investigación y definición necesarias antes de comenzar a desarrollar. En esta etapa, también nos conocimos como equipo y la forma en que trabaja cada uno, pudiendo así entender cómo planificar de la manera más adecuada el desarrollo.

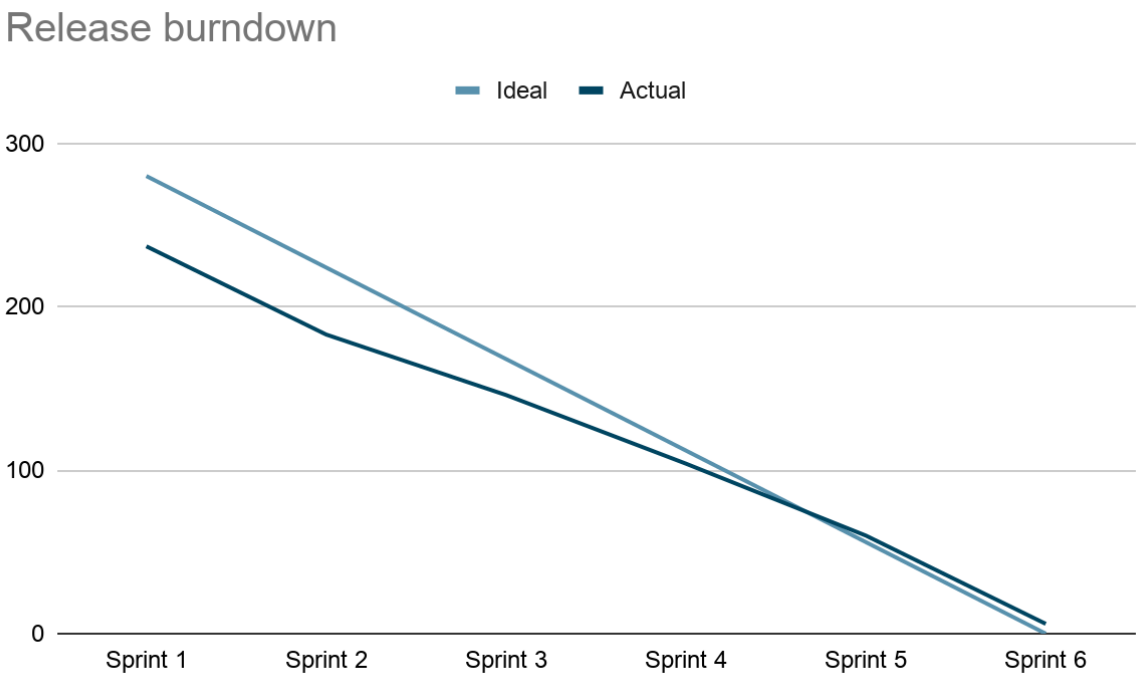
Para definir el largo de los *sprints* se tomaron en cuenta varios aspectos, el tiempo que cuenta cada integrante para desarrollar diariamente, la dificultad de la implementación de las funcionalidades y lo necesario para desarrollarlas con calidad, pruebas de estas. Se definieron entonces *sprints* de 2 semanas, siendo lo necesario para que en cada uno se pueda desarrollar de manera completa.

Inicialmente se definió una dedicación semanal de 15 horas por persona. Luego pasamos a definir cuánto esfuerzo corresponde un story point, concluyendo en 46 *story points* por *Sprint*.

Se completaron una totalidad de 6 *sprints*. Debajo se detalla la tabla mencionada.

Sprint	Tareas realizadas	Horas promedio por persona por semana	Velocidad (Story Points por Sprint)
0	→ Organización del equipo →Gestión de Riesgo →Planificación → Estudio del prototipo →Investigación de herramientas para el desarrollo → Análisis de requerimientos → Evaluación de tecnologías → Definición de prácticas de calidad	15.6	
1	→ Creación de repositorios → Modelado de tablas → Modelado de arquitectura → Estudio tecnologías → Desarrollo de los tickets de cada *sprint* → Visitas al cliente → Pruebas de concepto → Arreglo de bugs → Rediseño en programa especializado → Aplicación del rediseño	16.6	43
2		20	54
3		12.5	37
4		13.5	42
5		21.3	44
6		19.3	54





Se concluyó el *release* con un atraso de 6 *story points*, esto se debió a que en el mes de febrero los integrantes del equipo realizaron evaluaciones en la facultad, lo cual retrasó más de lo esperado. Se dejaron las tareas menos influyentes teniendo

en consideración el primer *MVP*, lo que de igual manera permitió culminar la primera iteración del ciclo evolutivo y luego cerrar el primer *release*.

4. Trabajo restante hasta el final

En cuanto a la terminal de autoservicio, se espera poder culminar con la integración con Mercado pago para que los clientes tengan la opción de pagar sus compras en la terminal si así lo desean, de lo contrario, se les brindara la opción de pagar en la caja antes de retirarse del bar.

Sobre la aplicación móvil, también será incorporada la integración con mercado pago y se debe incorporar las *notificaciones push* para que el usuario vaya a buscar su pedido cuando esté listo.

En la parte administrativa se creará una pantalla de *dashboards* para que los dueños puedan visualizar la analítica de los datos.

También se expondrán endpoints para facilitar la integración con terceros.

Además de esto, queda realizar reuniones con expertos, y el cliente para validaciones en las cuales pueden surgir cambios o nuevos requerimientos.

5. Proyección hasta el final del proyecto

Para el final del proyecto se espera tener un software funcional y responsive que abarque no solo las funcionalidades requeridas al comienzo del proyecto por el cliente, sino que también todas las que fueron surgiendo a lo largo del proyecto cada vez que se presentaba un *MVP*.

A continuación, presentaremos un roadmap con el fin de comunicar como fueron planeado los *sprints* y *releases* y cuáles quedan por delante. Tener en cuenta, que al estar trabajando en un ciclo de vida evolutivo, esta planificación se puede ver afectada ya que pueden ir surgiendo nuevos requerimientos o cambios en el sistema.

Hasta el momento, se viene trabajando con una velocidad promedio de 45 *story points* por *sprint*. Estimamos que esta velocidad se va a mantener o incluso

aumentar debido a la experiencia que el equipo va adquiriendo sobre las tecnologías y el dominio del problema.

- **Sprint 0 (1 de octubre al 30 de diciembre)**
- **1er release (11 de enero al 4 de abril)**
    - **Sprints**
        - Sprint 1 (11 de enero al 24 de enero)
        - Sprint 2 (25 de enero al 7 de febrero)
        - Sprint 3 (8 de febrero al 21 de febrero)
        - Sprint 4 (22 de febrero al 7 de marzo)
        - Sprint 5 (8 de marzo al 21 de marzo)
        - Sprint 6 (22 de marzo al 4 de abril)
- **2o release (5 de abril al 13 de junio)**
    - **Sprints**
        - Sprint 7 (5 de abril al 18 de abril)
        - Sprint 8 (19 de abril al 2 de mayo)
        - Sprint 9 (3 de mayo al 16 de mayo**)**
        - Sprint 10 (17 de mayo al 30 de mayo)
        - Sprint 11 (31 de mayo al 13 de junio)
- **3er release (14 de junio al 8 de agosto)**
    - **Sprints**
        - Sprint 12 (14 de junio al 27 de junio)
        - Sprint 13 (28 de junio al 11 de julio)
        - Sprint 14 (12 de julio al 25 de julio)
        - Sprint 15 (26 de julio al 8 de agosto)
- **Documentación y arreglos finales (9 de agosto al 30 de septiembre)**

12.3. Encuestas

Como parte de las actividades realizadas para recabar requerimientos se realizó una encuesta a personas que encajaran con el perfil de los usuarios del bar. A continuación, se pueden ver las preguntas realizadas y las respuestas obtenidas.

Experiencia de compra en un bar cervecero

Form description

Habitualmente concurre a bares cerveceros? *

- Nunca
- 1 o 2 veces por mes
- Más de 3 veces por mes

Concurre al mismo bar regularmente ? *

- Voy al mismo
- Prefiero ir rotando
- No me cambia ir al mismo o diferentes

Qué le parece el servicio generalmente en estos bares ? *

1. Muy lentos
2. Cuándo no hay mucha gente bueno, sino es muy lento
3. Siempre me parecen rápidos

Ha hecho alguna vez un pedido a través de un totem (pantalla táctil para hacer ordenes)? *

- ○ Sí
- ○ No
- ○ No recuerdo

Ha visto bares en los cuales los pedidos se realicen a través de un menú en el celular o a través de totems ? *

- ○ Si, entre 1-5 bares
- ○ Si, 5 o mas bares
- ○ No, nunca vi bares con esa modalidad

Se siente cómodo realizando una orden a través de una pantalla (celular o tótem) ? *

- Sí
- No
- Nunca realice alguna

Prefiere hacer una fila para que una persona le tome su orden o evitaría la misma y realizaría su pedido a través de un totem *

- Usaría el tótem
- Haria fila para que me atienda una persona

Al realizar una orden en el totem, prefiere pagar en el mismo o se siente mas cómodo pagando antes de retirarse en la caja o a un mozo. *

- Pagar en el totem
- Pagar en caja o al mozo antes de retirarme del bar
- Me da igual

Juegos en una aplicación

La aplicación constaría de juegos en los cuales los clientes podrán jugar y obtener descuentos en el bar, como también juegos en los cuales tendrán que interactuar con personas al azar del bar con el objetivo de conocer gente nueva.

Le gustaría jugar en el bar mientras espera su pedido o en cualquier otro momento ? *

- Sí
- No
- Prefiero solo charlar

Como cliente, le agregaría valor que el bar tenga una aplicación descargable con funcionalidades como juegos interactivos entre mesas *

- Si
- No
- Me da igual

De qué manera cree que le agrega valor?

Long answer text

Le parece muy molesto el hecho de tener que descargar una aplicación *

- Si
- No

Que funcionalidades extras le gustaría que la aplicación tuviese? *

Long answer text

Compartiría su información (perfil instagram) con otros participantes del juego? *

- Sí
- No
- Tal vez

Le gustaría tener una aplicación que facilite conocer gente dentro del bar? *

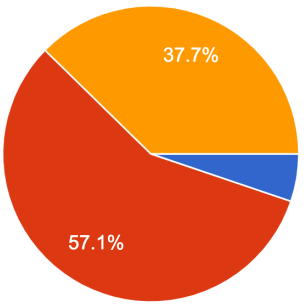
- Sí
- No
- Tal vez

Figura 82: Preguntas encuesta

Respuestas

Habitualmente concurre a bares cerveceros?

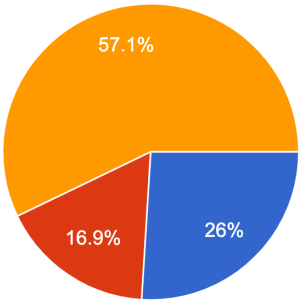
77 responses



Nunca
1 o 2 veces por mes
Más de 3 veces por mes

Concurre al mismo bar regularmente ?

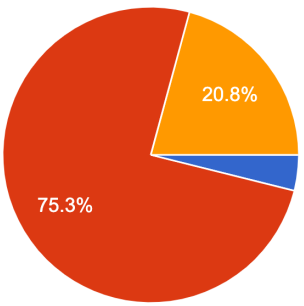
77 responses



Voy al mismo
Prefiero ir rotando
No me cambia ir al mismo o diferentes

Qué le parece el servicio generalmente en estos bares ?

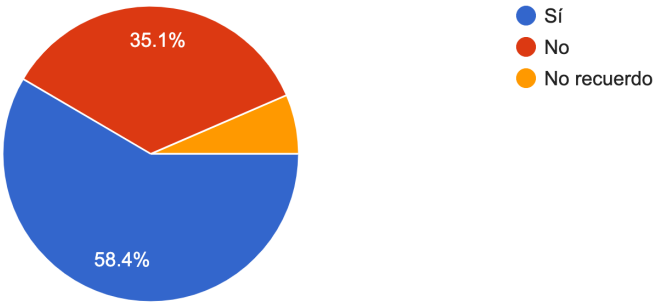
77 responses



Muy lentos
Cuándo no hay mucha gente bueno, sino es muy lento
Siempre me parecen rápidos

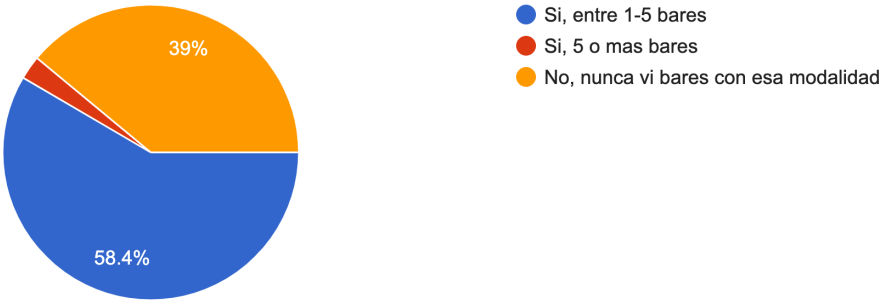
Ha hecho alguna vez un pedido a través de un totem (pantalla táctil para hacer ordenes)?

77 responses



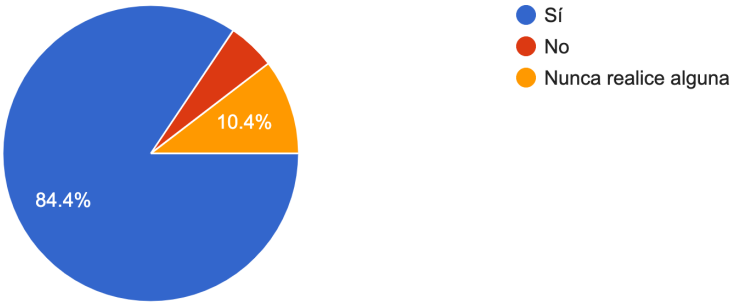
Ha visto bares en los cuales los pedidos se realicen a través de un menú en el celular o a través de totems ?

77 responses



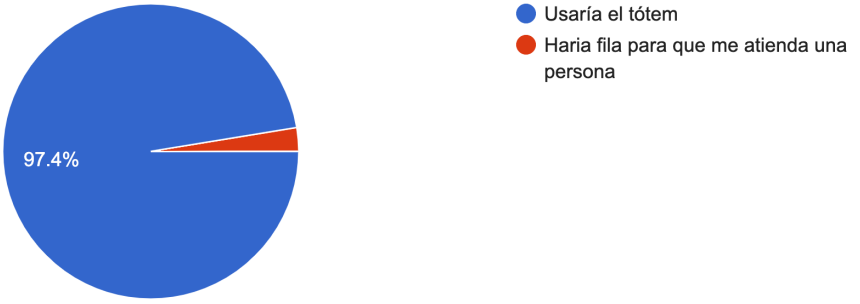
Se siente cómodo realizando una orden a través de una pantalla (celular o tótem) ?

77 responses



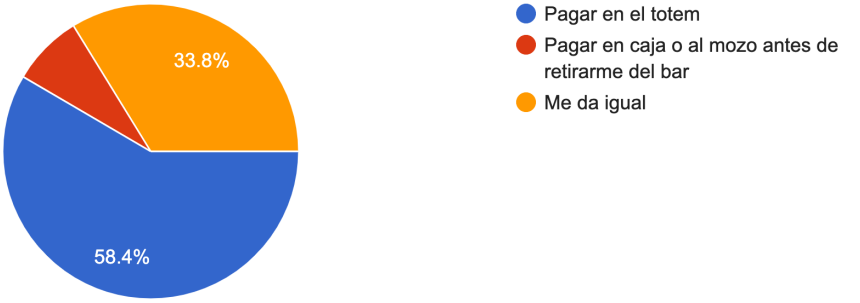
Prefiere hacer una fila para que una persona le tome su orden o evitaría la misma y realizaría su pedido a través de un totem

77 responses



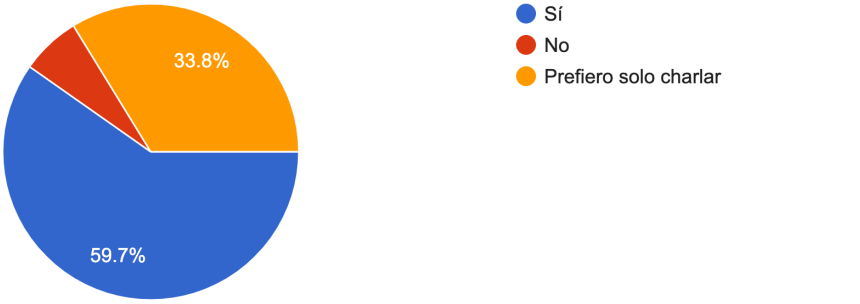
Al realizar una orden en el totem, prefiere pagar en el mismo o se siente mas cómodo pagando antes de retirarse en la caja o a un mozo.

77 responses



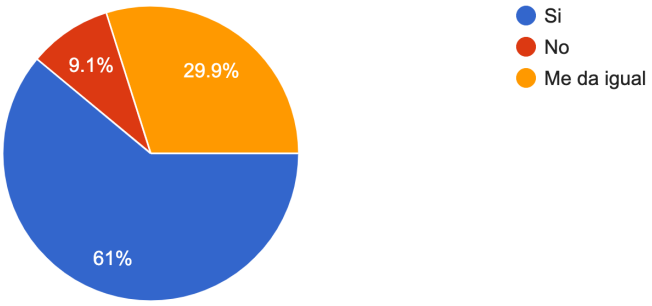
Le gustaría jugar en el bar mientras espera su pedido o en cualquier otro momento ?

77 responses



Como cliente, le agregaría valor que el bar tenga una aplicación descargable con funcionalidades como juegos interactivos entre mesas

77 responses



De qué manera cree que le agrega valor?

30 responses

Entretenimiento

Fidelizacion de clientes

Más divertido

Conocer gente nueva.

Te da un puntapié para iniciar ronda de interacciones entre tus acompañantes y te da el plus de un posible descuento, necesario estos días. Darían ganas de volver si funciona correctamente

Brinda entretenimiento para la espera

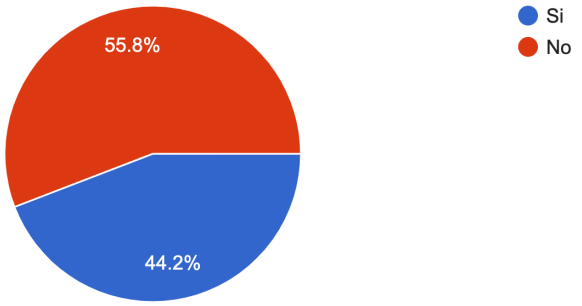
Es una forma adicional de entretenimiento y conocer personas fácilmente

Dinamismo y la posibilidad de obtener descuentos mediante los juegos

Lo haría más interactivo

Le parece muy molesto el hecho de tener que descargar una aplicación

77 responses



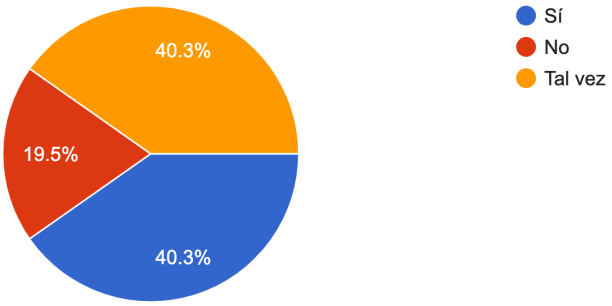
Que funcionalidades extras le gustaría que la aplicación tuviese?

77 responses

Ninguna
No se
El menú y formas de pago
Tinder
Juegos para jugar con las otras mesas y así interactuar
Yoqc
Juegos, menú, promociones
No lo se
D

Compartiría su información (perfil instagram) con otros participantes del juego?

77 responses



Le gustaría tener una aplicación que facilite conocer gente dentro del bar?

77 responses

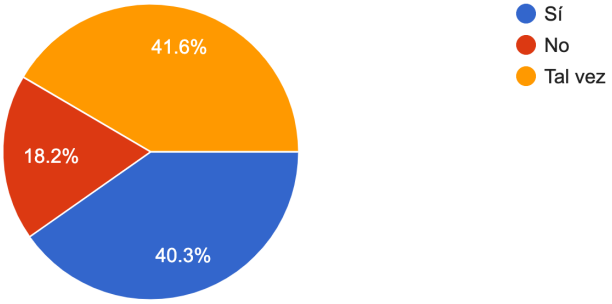


Figura 83: Respuesta encuesta

12.4. Graficas de gestión de alcance

12.4.1. Release 1 Burndown chart

A continuación, se presenta la tabla de velocidad del equipo real y la esperada por *sprint*, también se presenta la gráfica correspondiente para este *release*.

	Ideal	Actual	Velocidad
Inicio	276	276	
Sprint 1	230	233	43
Sprint 2	184	179	54
Sprint 3	138	142	37
Sprint 4	92	100	42
Sprint 5	46	56	44
Sprint 6	0	2	54

Tabla 35: Release 1 Burndown.

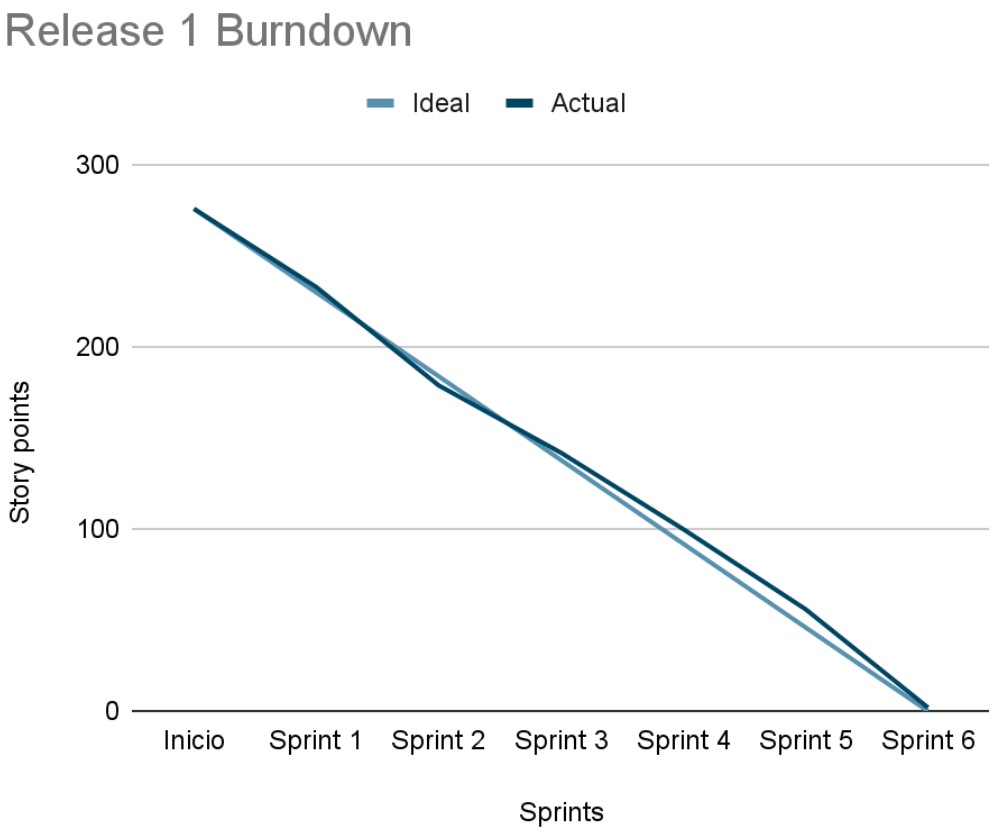


Figura 84: Release 1 Burndown chart

12.4.2. Release 2 Burndown chart

A continuación, se presenta la tabla de velocidad del equipo real y la esperada por *sprint*, también se presenta la gráfica correspondiente para este *release*.

	Ideal	Actual	Velocidad
Inicio	230	230	
Sprint 7	184	180	50
Sprint 8	138	132	48
Sprint 9	92	86	46
Sprint 10	46	39	47
Sprint 11	0	-9	48

Tabla 36: Release 2 Burndown.

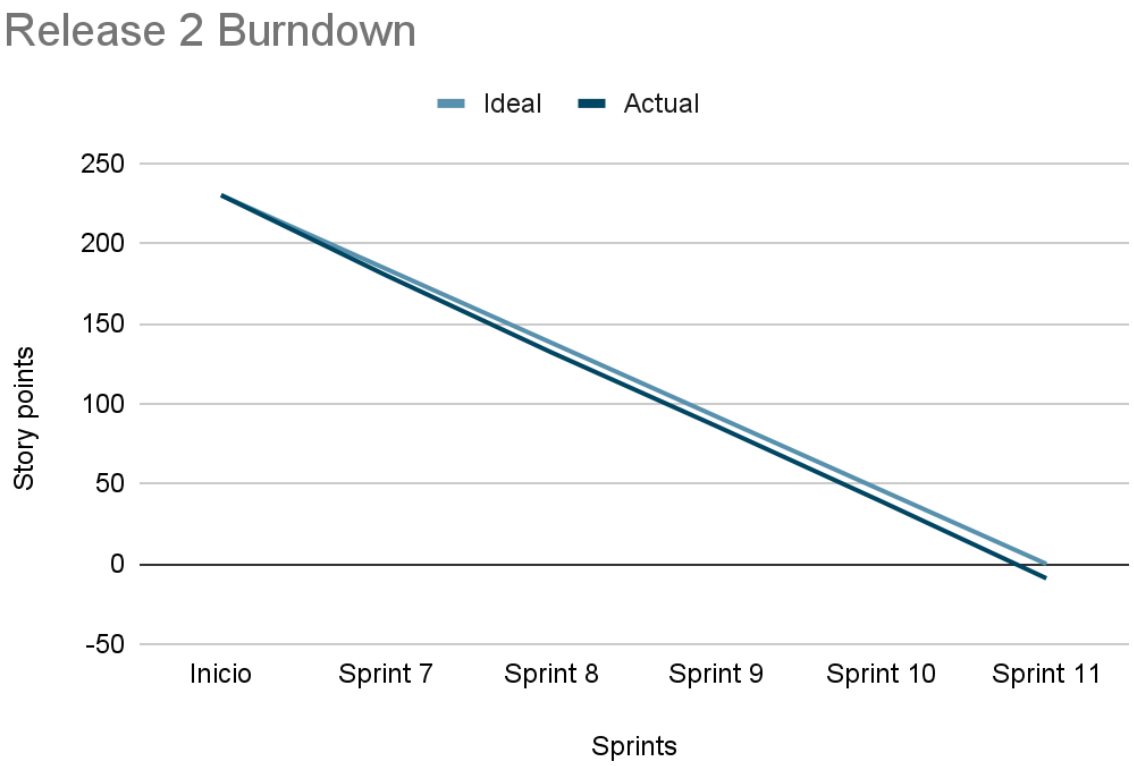


Figura 85: Release 2 Burndown chart

12.4.3. Release 3 Burndown chart

A continuación, se presenta la tabla de velocidad del equipo real y la esperada por *sprint*, también se presenta la gráfica correspondiente para este *release*.

	Ideal	Actual	Velocidad
Inicio	184	185	
Sprint 12	138	138	47
Sprint 13	92	96	42
Sprint 14	46	55	41
Sprint 15	0	2	53

Tabla 37: Release 3 Burndown.

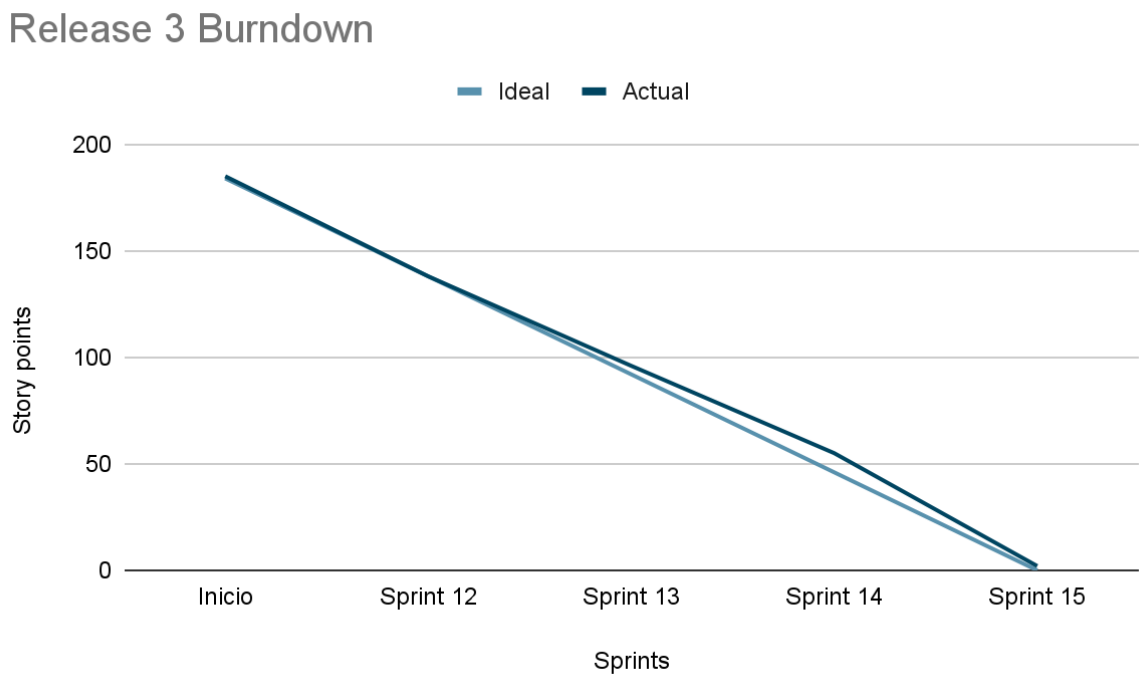


Figura 86: Release 3 Burndown chart

12.4.4. Incidentes

A continuación, se presenta la gráfica de incidentes registrados por *sprint* separados por severidad con la gráfica correspondiente.

	Leve	Medio	Severo
Sprint 1	0	0	0
Sprint 2	0	0	0
Sprint 3	4	1	0
Sprint 4	4	2	0
Sprint 5	9	3	0
Sprint 6	3	0	0
Sprint 7	7	2	0
Sprint 8	5	3	0
Sprint 9	4	3	0
Sprint 10	2	3	0
Sprint 11	4	2	0
Sprint 12	3	1	0
Sprint 13	3	0	0
Sprint 14	1	0	0
Sprint 15	0	0	0

Tabla 38: Incidentes.

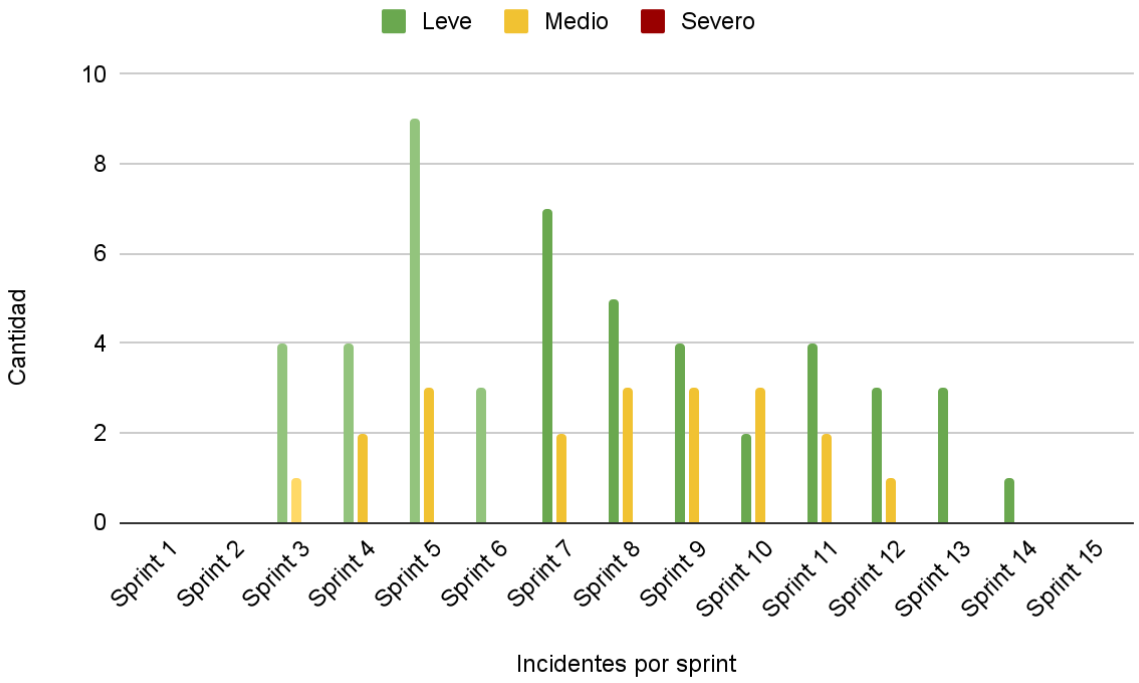


Figura 87: Incidentes por Sprint

12.4.5. Desvío

A continuación, se presenta la tabla del desvío de *story points* por *sprint* y la gráfica correspondiente.

	Desvío
Sprint 1	-3
Sprint 2	8
Sprint 3	-9
Sprint 4	-4
Sprint 5	-2
Sprint 6	8
Sprint 7	4
Sprint 8	2
Sprint 9	0
Sprint 10	1
Sprint 11	2
Sprint 12	1
Sprint 13	-4
Sprint 14	-5
Sprint 15	7

Tabla 39: Desvío.

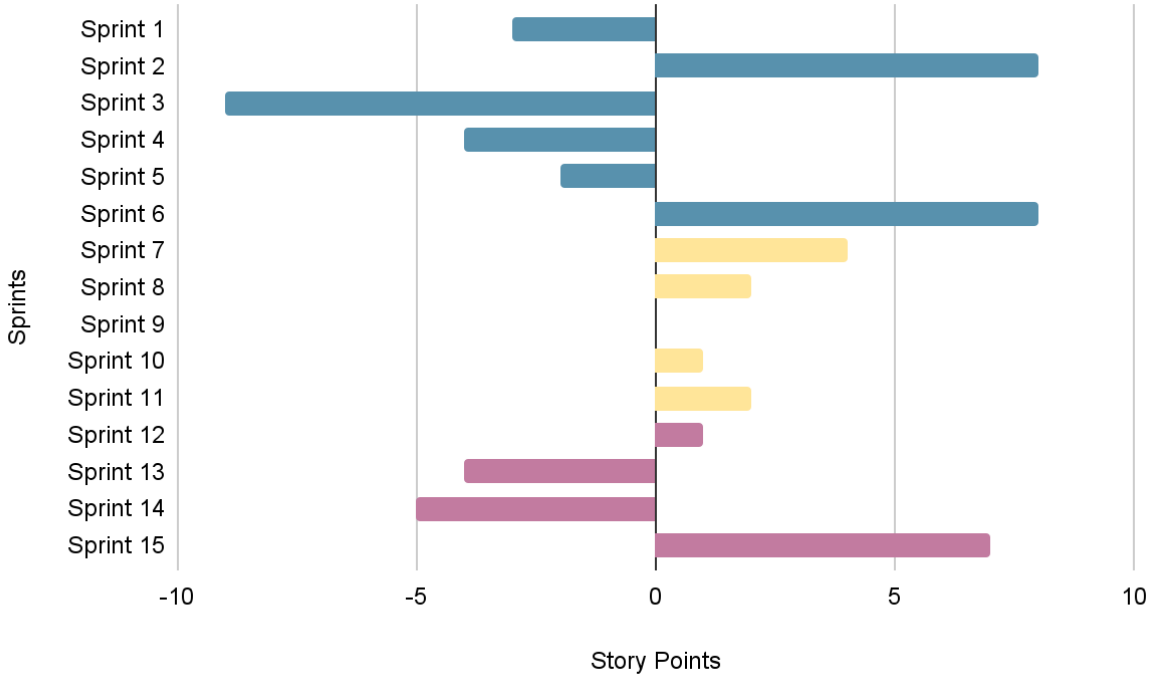


Figura 88: Desvío por Sprint

12.4.6. Horas promedio

A continuación, se presenta la tabla con las mediciones de horas por *sprint* dedicadas por cada integrante del equipo.

	Juan	Seba	Rodri	Horas Promedio
Sprint 1	34	34	32	33,33333333
Sprint 2	44	32	44	40
Sprint 3	33	12	30	25
Sprint 4	32	32	17	27
Sprint 5	42	44	42	42,66666667
Sprint 6	38	40	38	38,66666667
Sprint 7	37	38	39	38
Sprint 8	33	32	31	32
Sprint 9	30	30	29	29,66666667
Sprint 10	29	31	31	30,33333333
Sprint 11	31	31	30	30,66666667

Sprint 12	31	29	29	29,66666667
Sprint 13	38	22	20	26,66666667
Sprint 14	37	21	19	25,66666667
Sprint 15	38	40	43	40,33333333

Tabla 40: Horas promedio.

12.4.7. Velocidad

A continuación, se presentan las mediciones de velocidad separadas por *sprint* y la gráfica de la misma.

	Velocidad	Velocidad promedio
Sprint 1	43	45,66666667
Sprint 2	54	
Sprint 3	37	
Sprint 4	42	
Sprint 5	44	
Sprint 6	54	
Sprint 7	50	47,8
Sprint 8	48	
Sprint 9	46	
Sprint 10	47	
Sprint 11	48	
Sprint 12	47	45,75
Sprint 13	42	
Sprint 14	41	
Sprint 15	53	

Tabla 41: Velocidad.

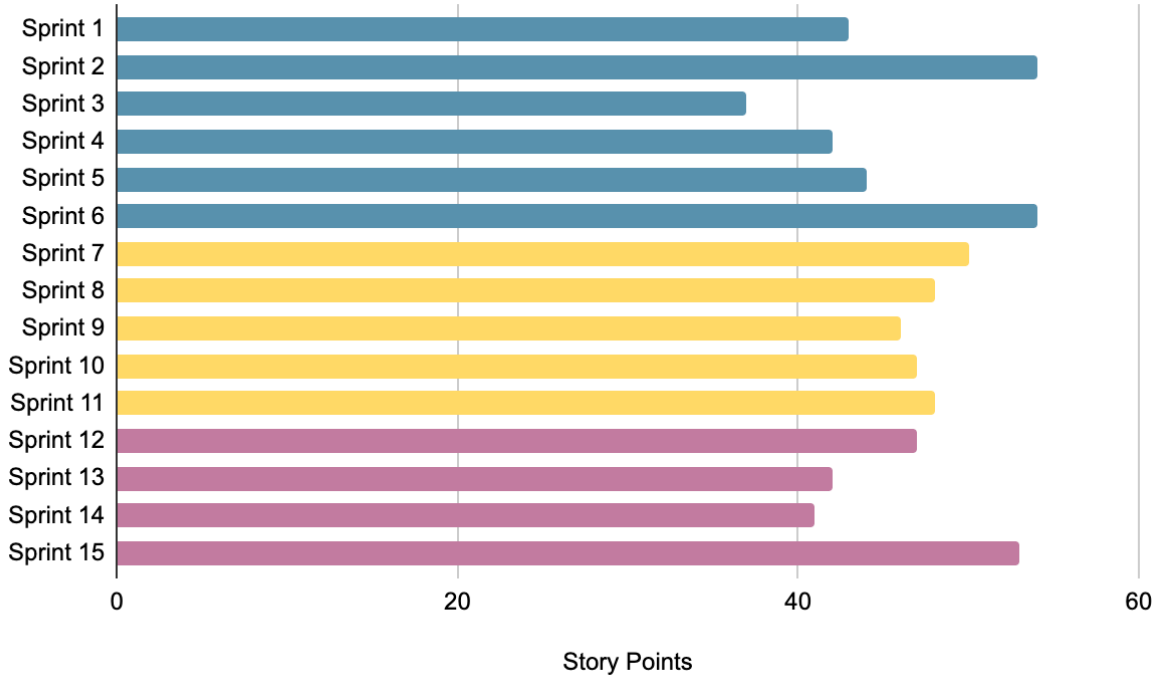


Figura 89: Velocidad por Sprint

12.4.8. Esfuerzo por área de trabajo

A continuación, se presentan las gráficas de la distribución del esfuerzo por área de trabajo separado por *release*.

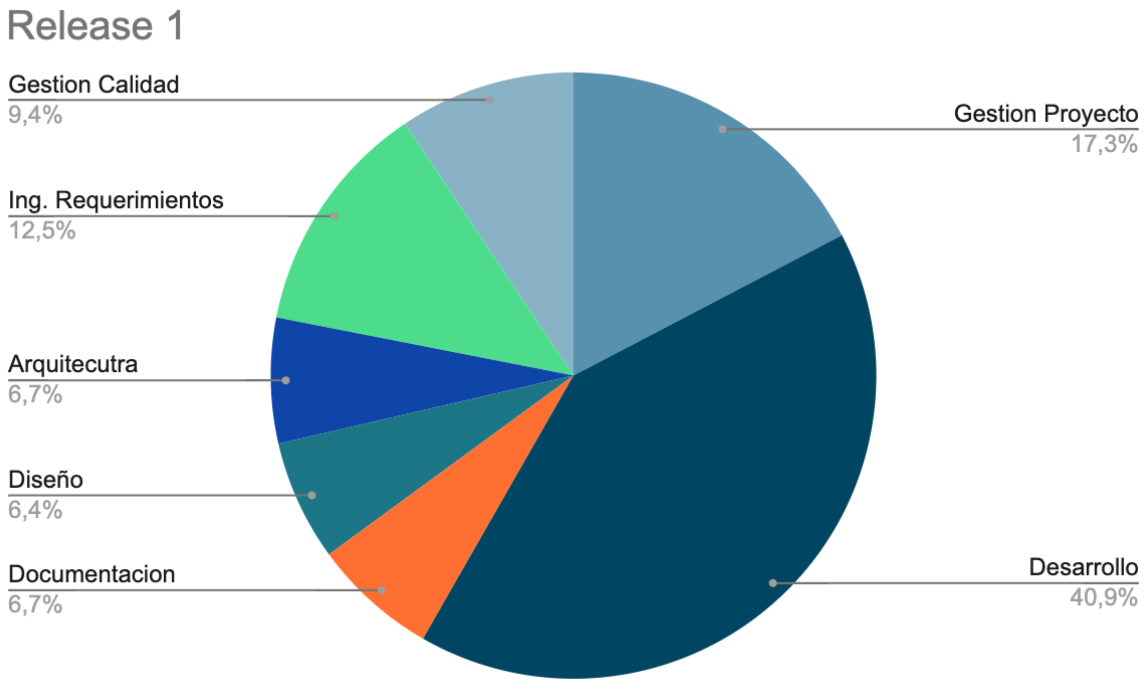


Figura 90: Esfuerzo por área de trabajo Release 1

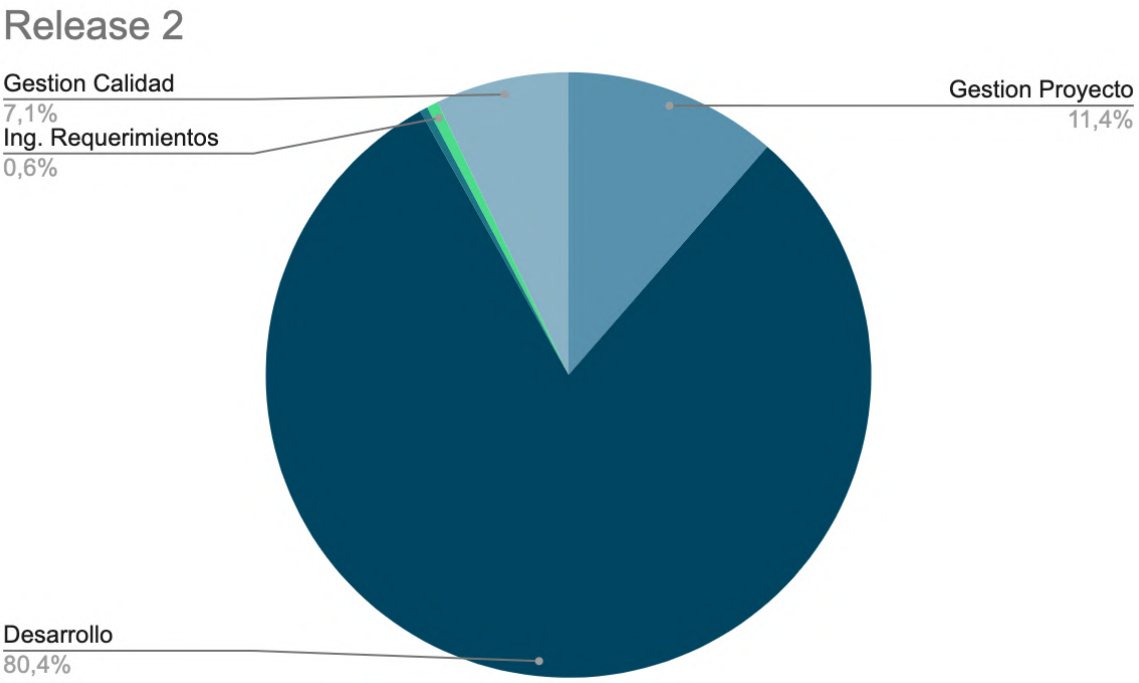


Figura 91: Esfuerzo por área de trabajo Release 2

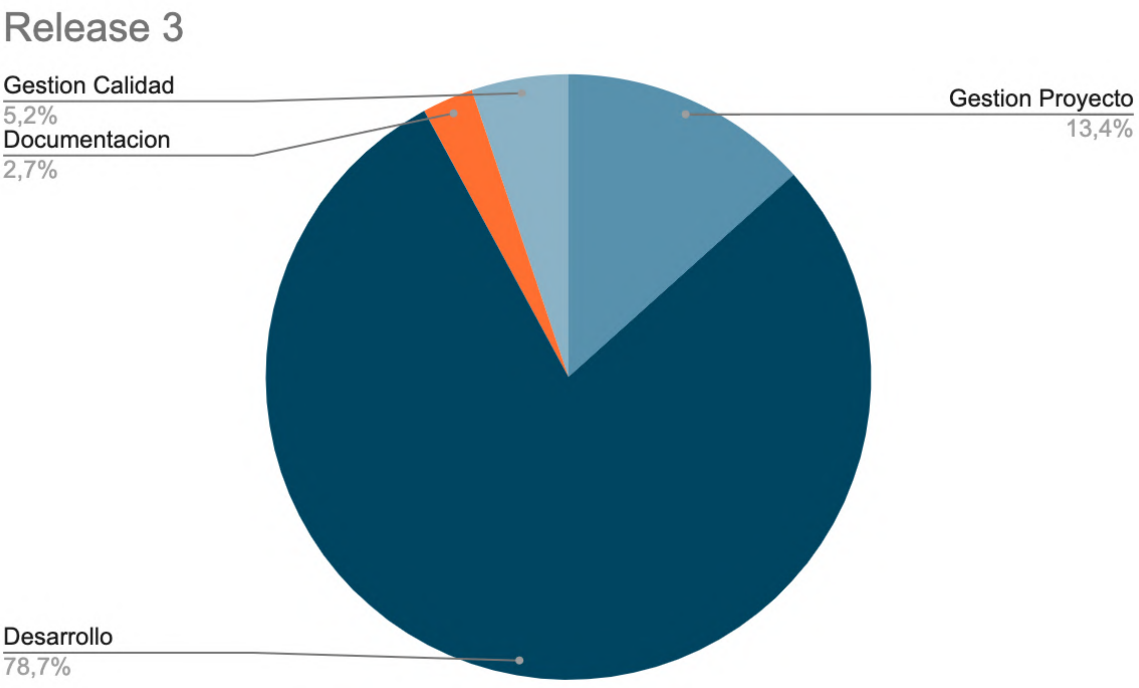


Figura 92: Esfuerzo por área de trabajo Release 3

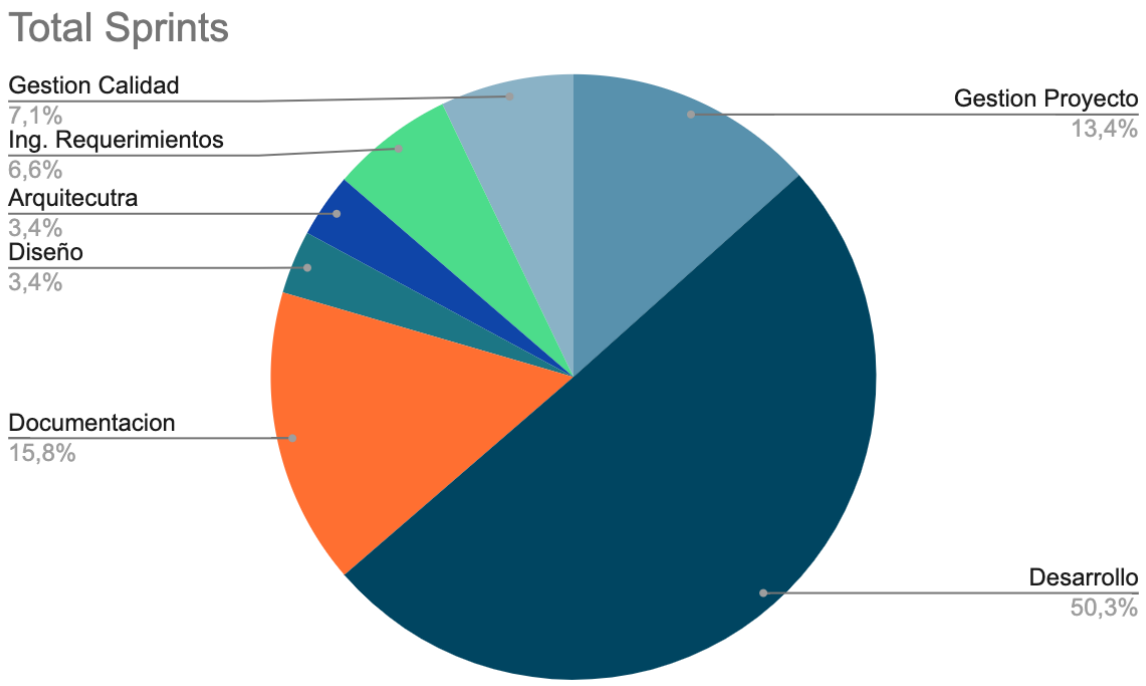


Figura 93: Esfuerzo por área de trabajo

12.5. Entrevistas

12.5.1. Entrevistas a usuarios

Preguntas

¿Qué esperas que se pueda hacer en la aplicación?

¿Cuál crees que sería la diferencia con la terminal autoservicio?

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

¿Qué opiniones le merece el prototipo?

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

Entrevista a Carla

¿Qué esperas que se pueda hacer en la aplicación?

Ver todas las opciones del local, con los detalles de cada producto. Que se puedan hacer los pedidos y pagar por ahí. También que existan promociones que solo sean

a partir de la aplicación, de esa forma me motivaría a bajarla y no me molestaría que me saque espacio del celular.

¿Cuál crees que sería la diferencia con la terminal de autoservicio?

Que lo tengas en la comodidad de tu celular, no tener que pararte, esperar a un mozo o lo que sea. También si se puede pagar desde la aplicación significa que no tengo que salir con plata o tarjeta lo que para mí es una molestia.

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

Lo ideal sería una notificación, no sé si con el tótem es posible, pero sino con los timbres que te dan en algunos lugares que suenan y vibran.

¿Qué opiniones le merece el prototipo?

Me gusta mucho, creo que es muy fácil e intuitivo clave para que lo pueda usar cualquier persona y que no sea algo negativo para el bar. Me parece clave que todo tenga nombre, precio, descripción y foto, me molesta ver un menú que solo algunas cosas tienen foto, en el caso de la comida.

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

El modelo tradicional es todo muy lento, dependes de un mozo o ir a hacer cola. Mirándolo desde el lado de la empresa, son menos empleados, menos sueldos por lo que debería generar una mayor rentabilidad y también evitas posibles errores. Del lado del consumidor creo que si una primera vez alguien te lo explica de forma clara es mucho más rápido y fácil. Creo que el rubro es muy competitivo y sin dudas estas facilidades hacen la diferencia a la hora de elegir dónde ir.

Entrevista a María

¿Qué esperas que se pueda hacer en la aplicación?

Espero que la aplicación me muestre el menú del bar, me permita realizar pedidos y pagarlos y poder ver mi historial de pedidos.

¿Cuál crees que sería la diferencia con la terminal de autoservicio?

Más ágil ya que no tengo que hacer cola para poder pedir, simplemente lo hago desde mi celular. A su vez supongo que el pago también se facilita porque en mi celular tengo las tarjetas guardadas.

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

En la app con una notificación. En el tótem lo veo más complicado, pero quizás con un número en el pedido y que haya una pantalla o con los cositos que dan ahora en muchos mercados que te llevas a la mesa y suenan.

¿Qué opiniones le merece el prototipo?

Me parece que está bueno, moderno, se entiende fácil. Me gustan los colores.

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

El modelo tradicional me parece que en bares no está bueno, siempre hay mucha fila, gente parada, y más ahora con el covid. El modelo de tótem me parece que está bueno para agilizar y aparte pueden poner la cantidad que sea necesaria. Lo que sí, si ponen solo uno "reemplazando" al empleado que toma los pedidos me parece que no serviría de mucho porque habría fila igual.

Entrevista a Gastón

¿Qué esperas que se pueda hacer en la aplicación?

Realizar pedidos de comida/bebidas. Reservas mesas para una noche, pagar la cuenta. Dividir cuentas.

¿Cuál crees que sería la diferencia con la terminal autoservicio?

Creo que sería un cambio bienvenido, agilizaría la toma de pedidos, evitaría errores humanos al anotar pedidos.

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

Vía App. Notificación de que el pedido está pronto y lo paso a buscar.

Vía totem. Lo mejor sería él un *beeper* que vibre cuando mi pedido está listo y puedo retirarlo

¿Qué opiniones le merece el prototipo?

Esta buena, creo que da poco lugar al error del usuario a utilizar la app, las cards de cada plato son grandes, claras, si las imágenes representan correctamente el plato me parece que está bueno, te ahorras de errarle a lo que pedir.

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

Capaz es la pandemia o simplemente el avance de la tecnología en lo cotidiano, pero creo es la evolución natural a seguir, esta aplicación quita fricción de realizar pedidos en locales y esto se traduce en clientes más felices y que disfrutan más de la noche

Entrevista a Luciana

¿Qué esperas que se pueda hacer en la aplicación?

De lo que entendí creo que voy a poder llegar al bar y pedir en una pantalla lo que quiero. También podría bajarme una aplicación para hacer lo mismo.

¿Cuál crees que sería la diferencia con la terminal autoservicio?

Me gusta la idea porque odio tener que hacer fila cada vez que quiero pedir un trago o algo para comer.

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

En el tótem no estoy segura, supongo que es como en algunos lados que te dan un aparato que hace ruido cuando está pronta la comida. Y en la aplicación también supongo que es como cualquier otra aplicación que te llega una notificación.

¿Qué opiniones le merece el prototipo?

Me gusta mucho la verdad, me parece bastante original.

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

No tengo una opinión ya pensada. El tradicional me imagino como cualquier otro bar esperando en la fila para pedir y yendo a buscar la comida después. Un bar con un tótem me lo imagino más moderno y actualizado, tal vez coincide con un bar a la moda.

Entrevista a Gonzalo

¿Qué esperas que se pueda hacer en la aplicación?

Espero poder hacer todo lo que puedo hacer en persona en el bar, comprar comida, bebidas, que me avisen cuando está pronto, llevar un traqueo de todo lo que compro y pido históricamente.

¿Cuál crees que sería la diferencia con la terminal autoservicio?

No estoy seguro que impacto puede llegar a generar. Supongo que agregando más puntos de atención el servicio va a mejorar sustancialmente.

¿Cómo te parece que te deberían avisar que el pedido se encuentra pronto (tótem y app)?

Eso lo puedo imaginar bastante fácil por otros bares donde ya vi tótems. Para tótems se usa un beeper y para la aplicación que me llegue un mensaje.

¿Qué opiniones le merece el prototipo?

En general me parece interesante, quizás le haría algunos cambios. Soy un poco exigente con las aplicaciones, me gusta que se entienda fácil. Mejoraría el tema de las categorías y en algo así casi todo debería poder hacerse en una sola pantalla.

¿Tienes alguna opinión al respecto sobre el modelo tradicional y el modelo de tótem?

Si, me considero una persona vanguardista con la tecnología. Si me preguntas a mi en un futuro me gustaría que la atención sea toda automatizada por pantallas y aparatos que me dejen atenderme solo.

12.5.2. Entrevista a empleados

Por empleados se entiende a los cocineros y a las personas que atienden en la barra

Preguntas

¿Qué problemas encuentra en el modelo de atención actual?
¿Los problemas planteados anteriormente (en caso de haberlos) cree que es algo común entre los bares o que es específico de este bar?
¿Qué le parece el prototipo?
¿Cree que la solución explicada facilita o favorece su trabajo?

Entrevista a Julieta - Cajera

¿Qué problemas encuentra en el modelo de atención actual?

Un problema muy grande que estamos teniendo es que yo trabajo en la barra tanto tomando pedidos en persona como los pedidos por WhatsApp, eso hace que tenga que estar pendiente del WhatsApp y si estoy atendiendo un pedido presencial demore o se me pase algún mensaje enviado. Otra cosa que nos pasa es que como los pedidos son enviados por mensaje pueden no estar claros y generar confusiones y hay que seguir hablando para corroborarlos.

¿Los problemas planteados anteriormente (en caso de haberlos) cree que es algo común entre los bares o que es específico de este bar?

Los problemas de atención son comunes en todos los bares debido a las horas pico, igual particularmente este bar tiene el problema que te comenté que se debe al uso de WhatsApp que creo que no hay otro que haga lo mismo.

¿Qué le parece el prototipo?

Me re gusta. Parece ser fácil de usar y creo que está toda la información en pantalla.

¿Cree que la solución explicada facilita o favorece su trabajo?

Si por lo que te comenté anteriormente, con esta solución me parece que muchos de los problemas que estamos teniendo actualmente como que se mezclen los mensajes no los tendríamos.

Entrevista a Sofía - Cocinera

¿Qué problemas encuentra en el modelo de atención actual?

En los momentos en los que hay muchos pedidos al mismo tiempo es difícil mantener un orden de cuales fueron pedidos antes y cuáles fueron pedidos después. También otro problema que nos pasa es que cuando quieren cancelar o modificar un pedido es difícil identificar cuál de todos los tickets es.

¿Los problemas planteados anteriormente (en caso de haberlos) cree que es algo común entre los bares o que es específico de este bar?

Trabajé en otro bar y teníamos los mismos problemas con las órdenes.

¿Qué le parece el prototipo?

Me gusta como quedo, creo que está bien pensado y me gustaría poder probarlo.

¿Cree que la solución explicada facilita o favorece su trabajo?

Como te dije creo que está bien pensado y creo que está bien pensado. Me parece que muchos de los problemas que te comenté se irían con esto.

Entrevista a María Noel - Cocinera

¿Qué problemas encuentra en el modelo de atención actual?

Se nos mezclan los pedidos y por lo tanto a veces la entrega de la comida pueda demorarse más de lo normal.

¿Los problemas planteados anteriormente (en caso de haberlos) cree que es algo común entre los bares o que es específico de este bar?

La verdad es que no sé, pero supongo que es de todos.

¿Qué le parece el prototipo?

Me gusto. Creo que puede ser útil sobre todo para cuando hay muchos pedidos juntos. Creo que a los clientes les va a encantar.

¿Cree que la solución explicada facilita o favorece su trabajo?

Con lo que me contaron sin duda que los pedidos se van a poder controlar mucho mejor y por lo tanto se van a entregar más rápido.

12.6. Diagrama de base de datos

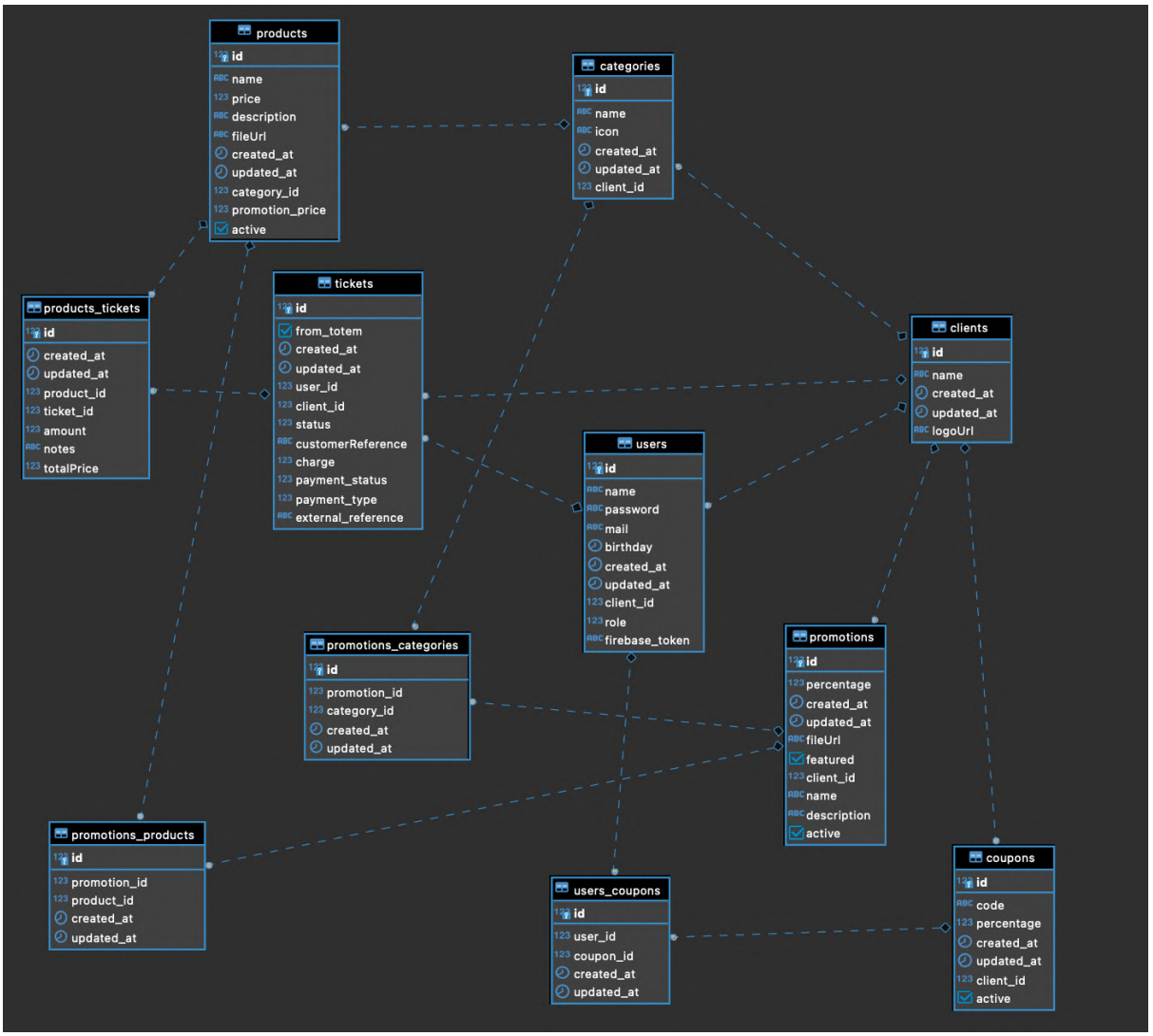


Figura 94: Diagrama de base de datos

12.7. Diagrama de interacción

Interacción para pedir una orden desde una terminal de autoservicio (Tótem)

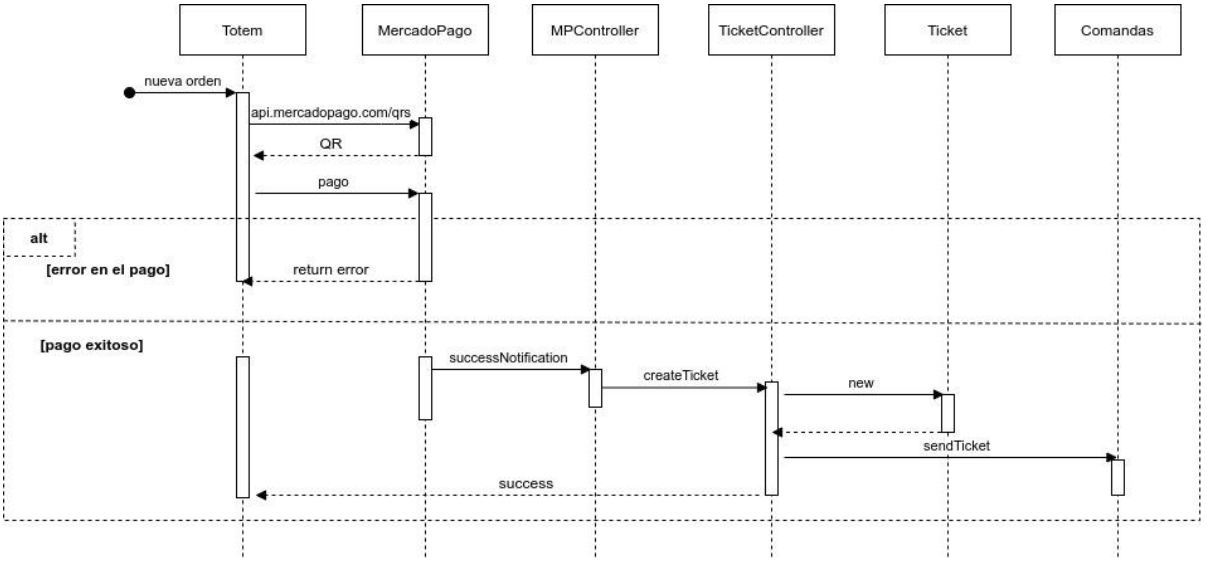


Figura 95: Diagrama de interacción orden desde totem

Luego de concretarse el pedido, la nueva comanda llega a la cocina. La siguiente es la interacción desde que un cocinero confirma la comanda como pronta para retirar.

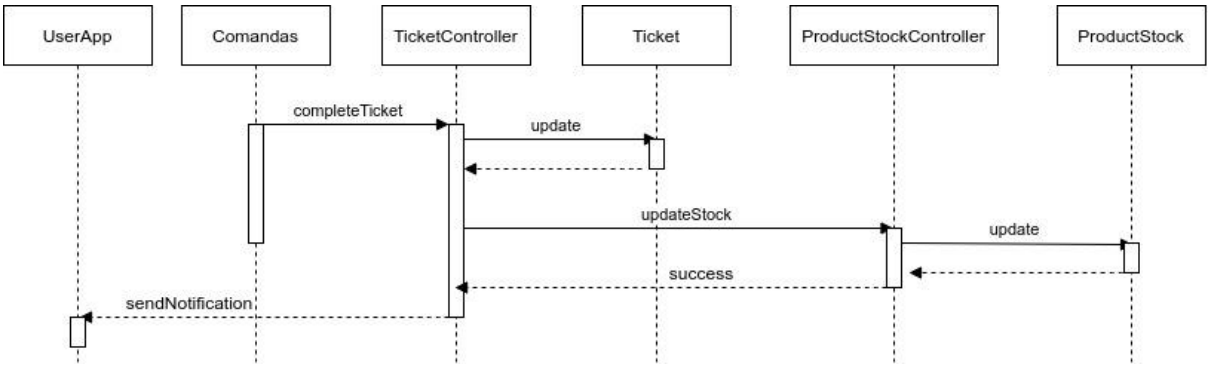


Figura 96: Diagrama de interacción confirmación de comanda

12.8. Plan de release

En el plan de *release* se puede ver para cuando se planificó realizar cada requerimiento funcional y cuando efectivamente fue llevado a cabo.

Ruby on Rails	Release 1		Release 2		Release 3	
	4 abril		13 junio		8 agosto	
	Plan	Real	Plan	Real	Plan	Real
RF1 Gestión de usuarios						
RF2 Gestión de productos						
RF3 Gestión de categorías						
RF4 Gestión de promociones						
RF5 Gestión de cupones						
RF6 Gestión de nuevo negocio						
RF7 Gestión de comandas						
RF8 Gestión de stock						
RF9 Gestión comandas en efectivo						
RF10 Mercado pago - Actualizar status pedido						
RF11 Pantalla dashboard						
RF12 Pantalla caja						
RF13 Envío de notificaciones						
RF14 Envío email al recuperar contraseña						

Tabla 42: Plan de Release backend.

Flutter	Release 1		Release 2		Release 3	
	4 abril		13 junio		8 agosto	
	Plan	Real	Plan	Real	Plan	Real
RF15 Login en aplicación						
RF16 Registro de usuario						
RF17 Recuperar contraseña						

RF18 Menú						
RF19 Agregar producto						
RF20 Carrito						
RF21 Selección de beeper						
RF22 Perfil de usuario						
RF23 Editar perfil de usuario						
RF24 Histórico de pedidos						
RF25 Repetir pedidos						
RF26 Pedidos sin pagar						
RF27 Comandas						
RF28 Pantalla de promociones						
RF29 Pago con Mercado pago						
RF30 Pago con efectivo						
RF31 Aplicar promociones						
RF32 Ingreso cupones						
RF33 Flutter web						
RF34 Filtro categorías						
RF35 Push notifications						

Tabla 43: Plan de Release frontend.

12.9. Pruebas de usabilidad (segundos y clicks)

Una de las métricas que se utilizaron para el producto fue medir el tiempo promedio para realizar las tareas. Es por esto que se realizaron pruebas al final de cada *release*. lo que se hizo fue pedir que realicen las mismas personas un flujo específico.

El flujo que se pedía que realicen era el siguiente: pedir una hamburguesa (scooby doo), un gin tonic y una corona. luego eliminar la corona y luego pagar en efectivo.

Una vez que el usuario termina el pedido se registra la cantidad de tiempo que le llevó realizar esta tarea.

Se eligió realizarlo con las mismas personas para poder chequear el progreso de la usabilidad del sistema una vez que se implementan nuevos requerimientos y las mejoras sugeridas.

Al mismo tiempo que realizaban estas pruebas, se iba contando la cantidad de clicks que estos usuarios tenían que realizar para completar la tarea.

12.9.1. Release 1

Usuario	Segundos	Clicks
1	37	21
2	39	19
3	36	22
4	40	20

Tabla 44: Pruebas de usabilidad release 1.

12.9.2. Release 2

Usuario	Segundos	Clicks
1	35	19
2	35	17
3	34	21
4	36	17

Tabla 45: Pruebas de usabilidad release 2.

12.9.3. Release 3

Usuario	Segundos	Clicks
1	28	13

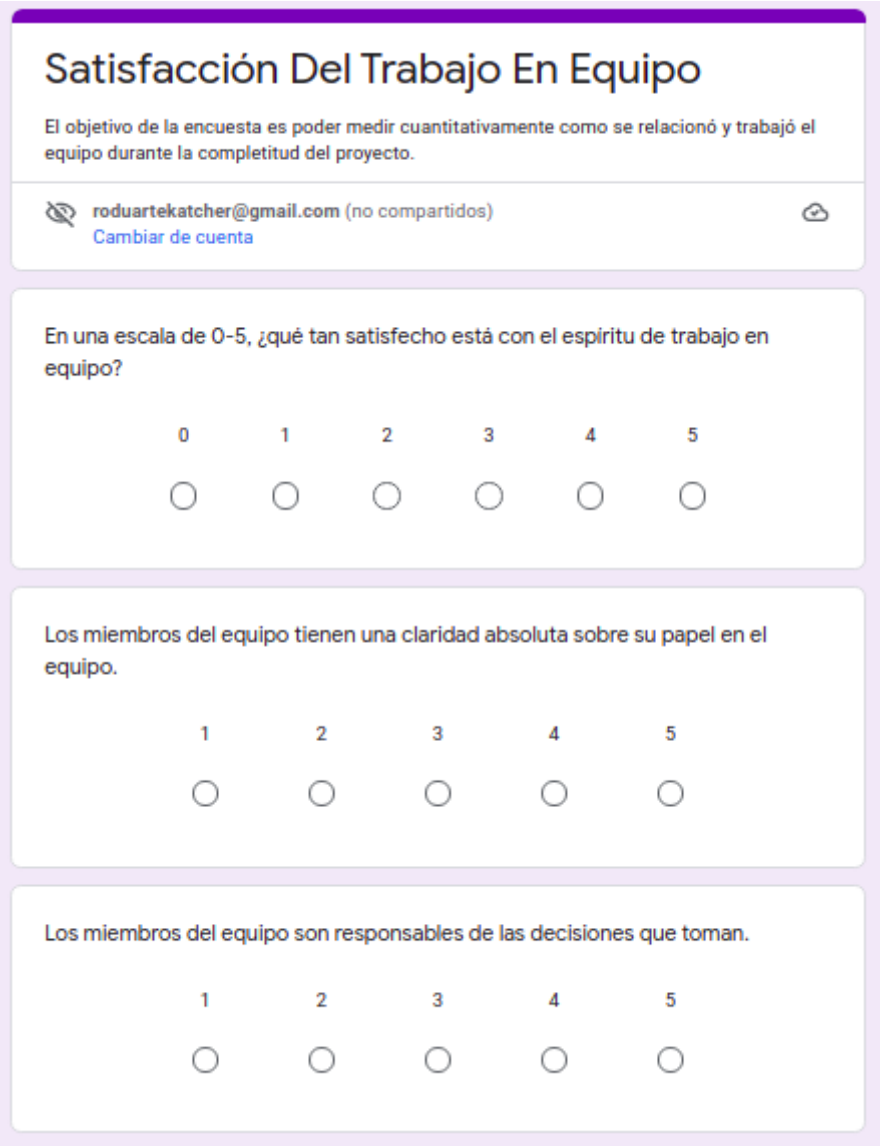
2	32	12
3	29	15
4	31	13

Tabla 46: Pruebas de usabilidad release 3.

Release	Segundos	Clicks
1	38	20
2	35	18
3	30	13

Tabla 47: Pruebas de usabilidad.

12.10. Encuesta de satisfacción del trabajo en equipo



Satisfacción Del Trabajo En Equipo

El objetivo de la encuesta es poder medir cuantitativamente como se relacionó y trabajó el equipo durante la completitud del proyecto.

roduartekatcher@gmail.com (no compartidos)
Cambiar de cuenta

En una escala de 0-5, ¿qué tan satisfecho está con el espíritu de trabajo en equipo?

0 1 2 3 4 5

Los miembros del equipo tienen una claridad absoluta sobre su papel en el equipo.

1 2 3 4 5

Los miembros del equipo son responsables de las decisiones que toman.

1 2 3 4 5

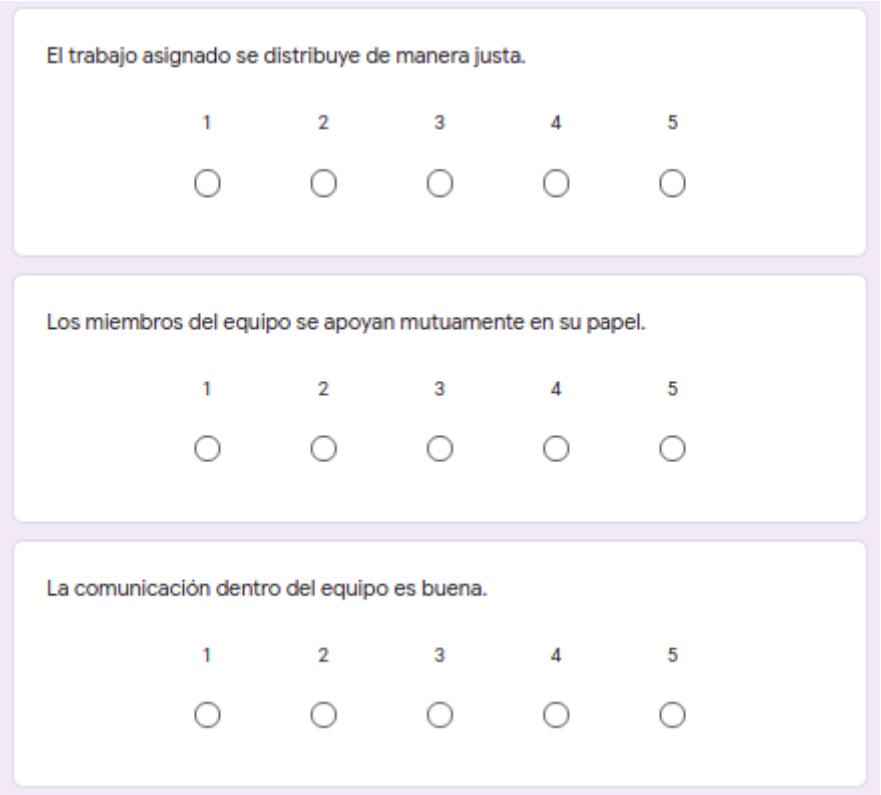
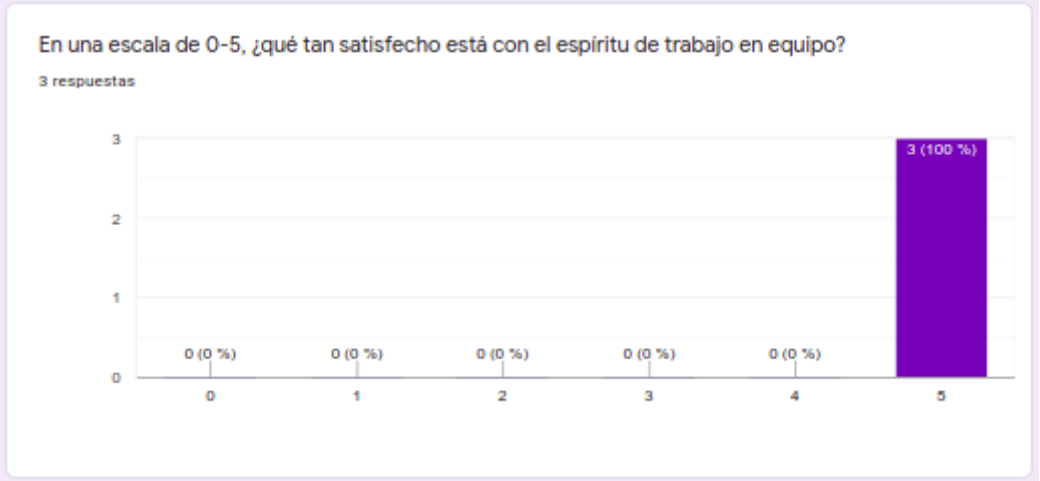
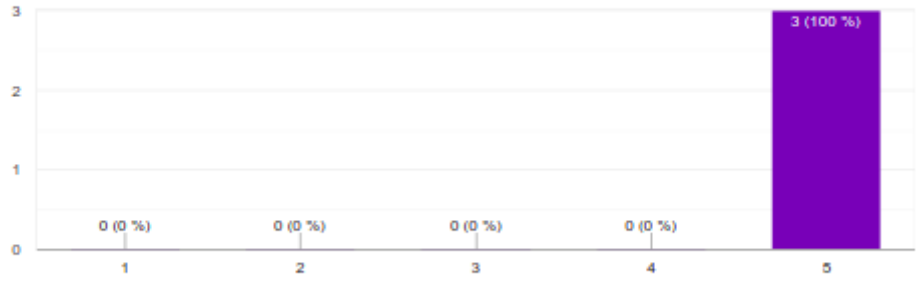


Figura 97: Encuesta Satisfacción del trabajo en equipo



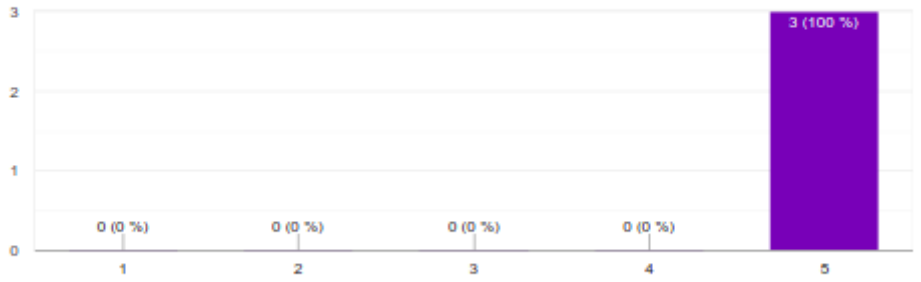
Los miembros del equipo tienen una claridad absoluta sobre su papel en el equipo.

3 respuestas



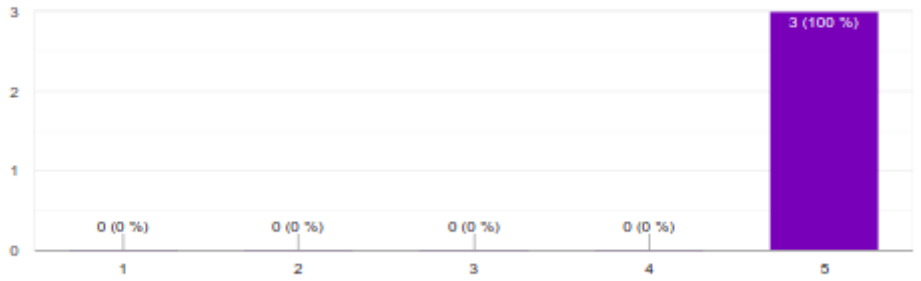
Los miembros del equipo son responsables de las decisiones que toman.

3 respuestas



El trabajo asignado se distribuye de manera justa.

3 respuestas



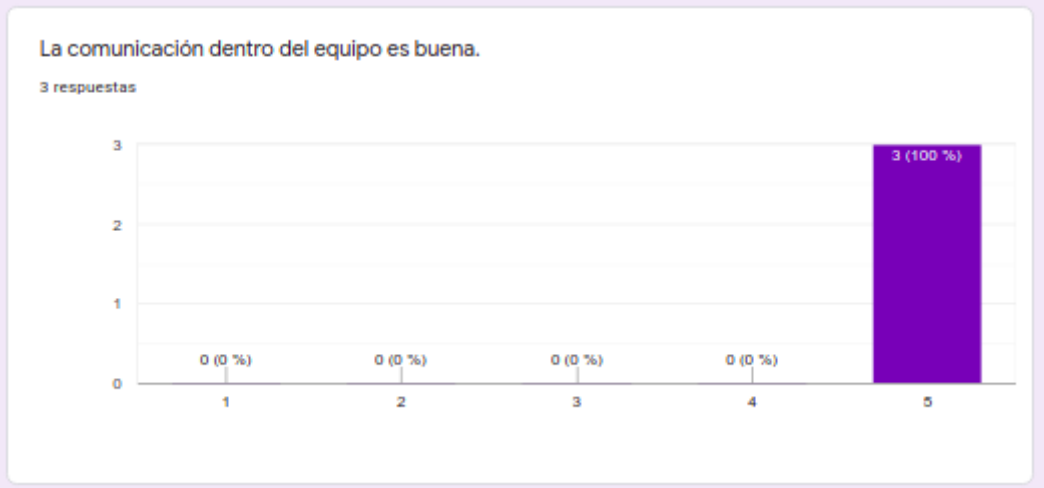


Figura 98: Respuesta de Encuesta Satisfacción del trabajo en equipo

12.11. Entrevista de satisfacción del cliente

¿Quedaron conformes con el trabajo realizado por el equipo?

“La verdad es que desde un principio se notó el compromiso que tenía el equipo con el proyecto y eso se vio reflejado en las reuniones que teníamos y lo que nos iban mostrando. Siempre que teníamos alguna duda o alguna nueva idea nos respondían rápido y para que nosotros pudiéramos entender. Como nosotras no sabemos mucho de tecnología pensábamos que iba a ser más difícil entender todo, pero terminó siendo sencillo por la dedicación y disposición brindada. El tiempo que llevamos probando la aplicación en el bar hemos tenido buena respuesta de las personas que lo usaron y en cocina también les fue fácil incorporarlo al día a día.”

¿Cree que el sistema desarrollado le será de utilidad para el futuro de su negocio?

“Por lo que llevamos probado creemos que viene siendo una buena experiencia y es posible que a futuro podremos usar el sistema completo. Si bien lo estamos usando de a poco nos gusta la idea de que todos los pedidos pasen por el mismo lugar y que a nosotros nos dé más información y control sobre nuestro negocio.”

¿Considera que fue buena la comunicación que tuvo el equipo con ustedes?

“La verdad que la disposición del equipo fue la mejor, siempre supieron responder nuestras dudas y ayudarnos en lo que necesitábamos. Nos quedamos con una imagen muy buena de este equipo y esperamos a futuro poder seguir trabajando con ustedes.”

12.12. Tabla de riesgos

Categoria	Nombre	Sprint 1			Sprint 2			Sprint 3			Sprint 4			Sprint 5		
		P	I	M	P	I	M	P	I	M	P	I	M	P	I	M
	Aprendizaje de la tecnologia	0.6	4	2.4	0.5	3	1.5	0.5	3	1.5	0.3	3	0.9	0.3	3	0.9
Tecnologicos	Riesgo plugin flutter	0.6	3	1.8	0.5	3	1.5	0.4	3	1.2	0.2	3	0.6	0.2	3	0.6
Equipo	Falta de compromiso integrante del equipo	0.2	3	0.6	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
	Mala relacion con el cliente	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
	Enfermedad de algun compañero	0.4	4	1.6	0.3	4	1.2	0.4	4	1.6	0.4	4	1.6	0.5	4	2
	Disponibilidad del equipo	0.5	4	2	0.4	4	1.6	0.9	4	3.6	0.4	4	1.6	0.4	4	1.6
	Existencia de problemas entre integrantes del equipo	0.2	3	0.6	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
Producto	Desconocer el dominio del producto	0.7	4	2.8	0.5	3	1.5	0.4	3	1.2	0.4	3	1.2	0.3	3	0.9
	Errores en la estimacion	0.6	3	1.8	0.5	4	2	0.7	4	2.8	0.5	4	2	0.5	4	2
	Abandono del cliente	0.3	4	1.2	0.3	4	1.2	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
Academicos	Abandono del tutor	0.2	5	1	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8

Figura 99: Tabla de riesgos (parte 1)

Sprint 6			Sprint 7			Sprint 8			Sprint 9			Sprint 10		
P	I	M	P	I	M	P	I	M	P	0.3	M	P	I	M
0.2	3	0.6	0.2	3	0.6	0.1	3	0.3	0.1	3	0.3	0.1	3	0.3
0.2	3	0.6	0.2	3	0.6	0.2	3	0.6	0.2	3	0.6	0.3	3	0.9
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
0.6	4	2.4	0.7	4	2.8	0.4	4	1.6	0.4	4	1.6	0.4	4	1.6
0.3	4	1.2	0.3	4	1.2	0.3	4	1.2	0.3	4	1.2	0.3	4	1.2
0.2	4	0.8	0.2	4	0.8	0.2	3	0.6	0.2	3	0.6	0.2	3	0.6
0.2	3	0.6	0.2	3	0.6	0.2	3	0.6	0.2	3	0.6	0.2	3	0.6
0.4	4	1.6	0.4	4	1.6	0.4	4	1.6	0.4	3	1.2	0.3	3	0.9
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.3	4	1.2
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8

Figura 100: Tabla de riesgos (parte 2)

Sprint 11			Sprint 12			Sprint 13			Sprint 14			Sprint 15		
P	I	M	P	I	M	P	I	M	P	I	M	P	I	M
0.1	3	0.3	0.1	3	0.3	0.1	3	0.3	0.1	3	0.3	0.1	3	0.3
0.4	3	1.2	0.3	3	0.9	0.2	3	0.6	0.3	3	0.9	0.3	3	0.9
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.3	5	1.5	0.2	5	1
0.4	4	1.6	0.4	4	1.6	0.4	4	1.6	0.4	4	1.6	0.4	4	1.6
0.7	4	2.8	0.8	4	3.2	0.8	4	3.2	0.4	5	2	0.4	5	2
0.2	3	0.6	0.2	3	0.6	0.2	3	0.6	0.2	4	0.8	0.2	4	0.8
0.2	3	0.6	0.2	3	0.6	0.1	3	0.3	0.1	3	0.3	0.1	3	0.3
0.3	3	0.9	0.2	3	0.6	0.2	3	0.6	0.1	3	0.3	0.1	3	0.3
0.3	4	1.2	0.3	4	1.2	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8
0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8	0.2	4	0.8

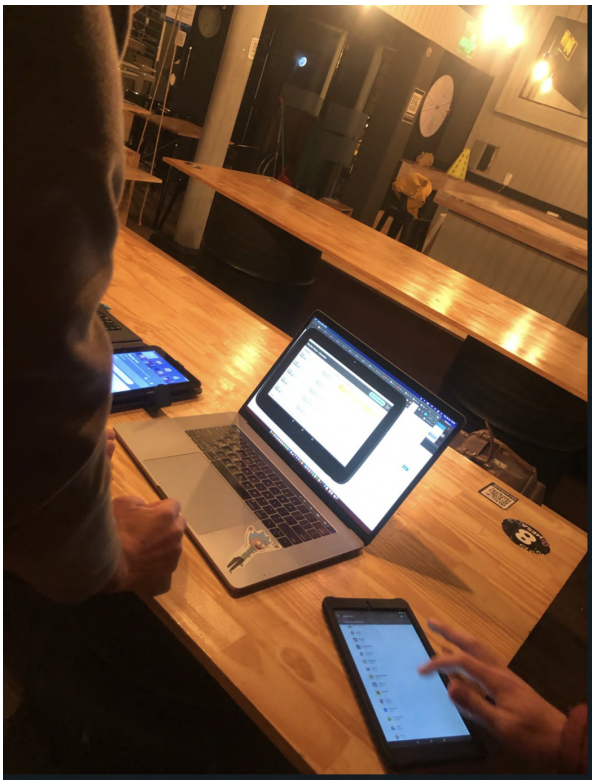
Figura 101: Tabla de riesgos (parte 3)

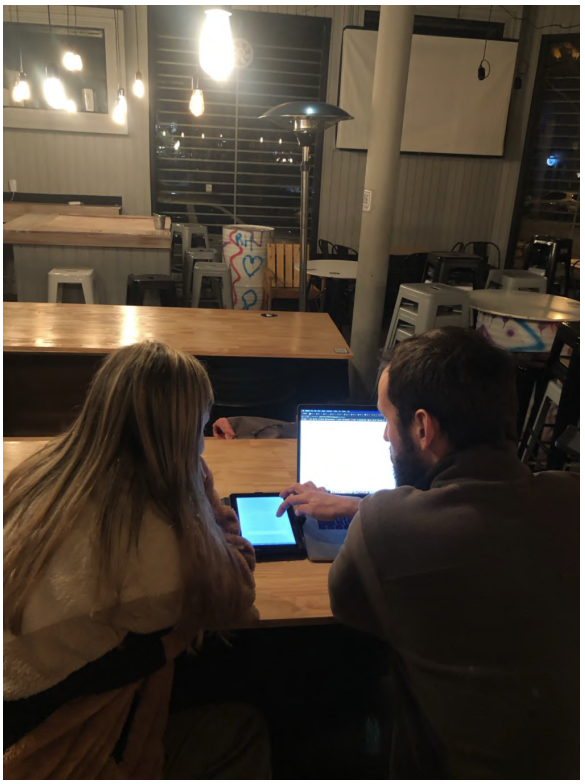
12.13. Horas promedio trabajadas

	Juan	Sebastián	Rodrigo	Horas Promedio	Story Points
Sprint 1	34	34	32	33,33333333	43
Sprint 2	44	32	44	40	54
Sprint 3	33	12	30	25	37
Sprint 4	32	32	17	27	42
Sprint 5	42	44	42	42,66666667	44
Sprint 6	38	40	38	38,66666667	54
Sprint 7	37	38	39	38	50
Sprint 8	33	32	31	32	48
Sprint 9	30	30	29	29,66666667	46
Sprint 10	29	31	31	30,33333333	47
Sprint 11	31	31	30	30,66666667	48
Sprint 12	31	29	29	29,66666667	47
Sprint 13	38	22	20	26,66666667	42
Sprint 14	37	21	19	25,66666667	41
Sprint 15	38	40	43	40,33333333	53

Tabla 48: Horas trabajadas

12.14. Fotos en el bar





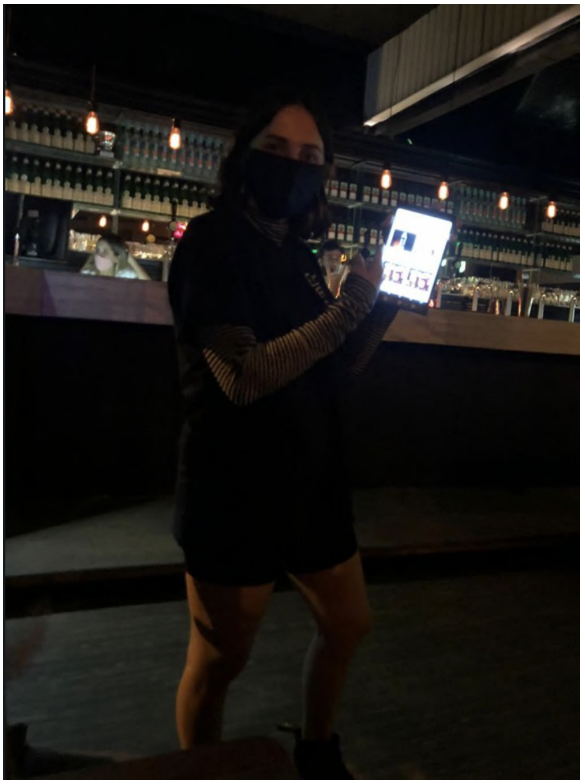


Figura 102: Fotos en el bar