Universidad ORT Uruguay

Facultad de Ingeniería

Astrolytics: Análisis de datos de mercado y generación de métricas de negocio

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

Federico Machado – 168803

Augusto Varela – 201190

Tutor: Ricardo Szyfer

2020

Declaración de autoría

Nosotros, Federico Machado y Augusto Varela, 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 realizábamos 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 qué fue contribuido por otros, y que fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Federico Machado

Augusto Varela

10 de Septiembre del 2020

10 de Septiembre del 2020

Agradecimientos

Federico Machado: Quiero agradecer a todos los que me han acompañado en mi paso por la ORT. Después de diez años es hora de finalmente cerrar mi etapa de estudiante en la universidad. Gracias a mi familia, amigos, docentes, tutores, compañeros de obligatorio, trabajo y de estudio que me han acompañado en este extenso trayecto a lo largo de mis dos carreras. Una mención especial a Carlos Sanguinetti y Álvaro Tasistro por darme todas las oportunidades que me dieron, y a Nora Szasz por la paciencia que me tuvo.

Augusto Varela: En primer lugar quiero agradecer a mi tutor, Ricardo Szyfer, por su orientación durante el proyecto. Logró guiar al equipo en situaciones de incertidumbre y motivarlo para sacar lo mejor del mismo. También agradezco a mi compañero, Federico Machado, por su buena disposición durante el proyecto y su paciencia ante mis faltas, permitiendome crecer como programador e ingeniero.

En segundo lugar quiero agradecer al personal del cliente, Astroselling, por su cooperación y disposición a colaborar con nosotros, ayudando a crear y desarrollar un producto valioso a través de un proceso disfrutable para ambas partes.

A mis amigos, Bryan, Carlos, Daniel, Ibrahim y Marco, agradezco por apoyarme durante este trayecto extenso de mi vida, estando ahí cuando más los necesitaba. También a Luis y Juan, por ser excelente compañía en varias noches de desvelo.

Finalmente agradezco a mi familia, por su paciencia con mis caprichos de programador y su apoyo durante estos meses de trabajo.

A todos ellos, muchas gracias.

Abstract

Astroselling es una plataforma de administración de stock, ventas y sincronización de *Enterprise Resource Planning* (ERPs), la cual trabaja sobre Mercado Libre y posee como cliente objetivo a medianas y grandes empresas que desean abordar el mercado digital.

La plataforma busca expandir sus funcionalidades y actualmente apunta a ofrecer datos y métricas del mercado donde sus clientes operan, para mejorar su toma de decisiones. De esta necesidad surge el concepto de Astrolytics, un módulo de análisis y generación de métricas de utilidad para los clientes de Astroselling.

El proyecto es llevado a cabo por dos estudiantes de ingeniería. Se utiliza una metodología de trabajo basada en SCRUM, la cual posee *sprints* de dos semanas y conlleva las ceremonias tradicionales, *sprint planning, sprint review y retrospective*.

El análisis y relevación de requerimientos fue ejecutada en base a entrevistas con Astroselling y sus clientes, creando así *user stories* que apuntan a agregar valor al negocio del consumidor.

Como medidas de control y aseguramiento de calidad se emplea el cumplimiento de buenas prácticas y seguimiento de estándares de codificación comunitarias como *StandardJS* y PEP 8, definiciones de atributos de calidad del ISO 9126, validaciones con usuarios utilizando betas y encuestas de satisfacción. A su vez, el desarrollo sigue un ciclo de vida basado en la metodología PDCA, con la cual luego de cada *sprint*, se ajusta el desarrollo para mejorarlo.

El producto consiste en un ecosistema de componentes separados por responsabilidades. Se poseen tres de ellos: Scraper, el cual se encarga de revisar el mercado digital y obtener datos para procesar. Está escrito en Python al igual que Analytics, el cual se encarga de procesar los datos de Scraper y generar reportes de valor para el usuario, y finalmente Commander, el cual funciona como un *frontend* para los consumidores y está desarrollado en Angular 8.

Como resultado, el equipo logró desarrollar un producto que conforma a los requerimientos iniciales, logró satisfacer las expectativas del cliente y ha sido

aceptado como producto empresarial, siendo ofrecido a clientes potenciales y existentes como punto de venta.

Palabras clave

Astrolytics, Astroselling, Métricas, Negocio, Mercado Libre, Angular, Python, Flask, Metodologías Ágiles

Índice

Glosario

Astrolytics: La plataforma a desarrollar. Engloba a todos los componentes de la solución.

Astroselling: Plataforma *Software as a service* (SaaS) que brinda varias herramientas y utilidades al consumidor de Mercado Libre para poder gestionar de manera más eficiente sus ventas. Entre estas herramientas se pueden encontrar sincronización de ERP, respuesta automática de preguntas y publicación masiva de productos.

Cliente: La plataforma Astroselling y la gente que la representa. Es el principal inversor de la plataforma y aporta requerimientos en base a lo que consigue de sus clientes.

Usuario: El consumidor final, quienes usan la plataforma de Astrolytics. Para el alcance de este proyecto, van a ser clientes de Astroselling o parte del equipo de Astroselling.

Tienda: Dentro del dominio de la aplicación, una tienda conforma una relación “n a n” con los usuarios. Estas se dan a entender como agrupaciones de transacciones y publicaciones dentro de Mercado Libre y Astroselling para n usuarios. Ergo, un usuario posee n tiendas, y una tienda puede ser accedida por n usuarios.

***Backend*:** “*Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos.“ [1]*

Frontend: *“Es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.” [1]*

***Scrapping*/Scrapeo:** Obtener información sobre una página o recursos remotamente a través de un proceso automatizado. Por ejemplo, nuestro componente Scraper obtiene información acerca de todas las publicaciones de Mercado Libre para que sean procesadas posteriormente.

Mercado Libre: "*Mercado Libre es una empresa dedicada a compras, ventas y pagos por Internet. Los usuarios pueden vender y/o comprar tanto productos nuevos como usados a un precio fijo o variable.*" [2]

PyMEs: Pequeñas y medianas empresas.

MVP: *Minimum Viable Product.* Para el contexto de este trabajo, se refiere a una plataforma mínimamente funcional que pueda entregarle valor a sus usuarios, quienes pueden acceder a ella externamente cuando lo deseen.

ERP: *Enterprise Resource Planning*. Con lo que respecta al proyecto, dígase ERP a una plataforma de software que utiliza el cliente para realizar operaciones en cuanto al inventario de sus productos, clientes y órdenes de compra. En particular se hace enfoque en la organización del stock, precio y otras posibles características de los productos como tìtulo, imàgenes, descripciòn, etc.

1. Introducción

1.1. Equipo de desarrollo

El equipo de desarrollo está conformado por dos integrantes de la carrera de Ingeniería en Sistemas, Federico Machado y Augusto Varela, bajo la tutoría del Ing. Ricardo Szyfer. La motivación del equipo de desarrollo consta principalmente en la culminación del trabajo final de la carrera.

También es importante mencionar que ambos se encuentran trabajando o en contacto con Astroselling al momento del comienzo del desarrollo, lo cual influenció ampliamente la elección del proyecto. Por otro lado, se tomó en cuenta la experiencia laboral de ambos integrantes, decantando por realizar un proyecto de desarrollo de software.

1.2. Cliente

Astroselling es una empresa fundada por Gonzalo Medeiros. Su plataforma homónima es un *Software as a Service* con varios planes de suscripción, cuyo objetivo es poder facilitar a sus clientes la gestión de canales de venta de todos sus productos, entre otras cosas.

En sus inicios, Astroselling tenía como único canal de venta Mercado Libre. Los usuarios podían realizar publicaciones masivas de una plataforma a otra. Posteriormente, se implementó la funcionalidad de sincronización con ERPs, de modo que si un producto se vendía en plaza, su stock se actualizaba en Mercado Libre automáticamente y viceversa, de modo que ya no era necesario tener que acceder a Mercado Libre ni estar al tanto de las modificaciones para poder operar correctamente.

Con el tiempo, Astroselling integró otros canales de venta como Shopify, WooCommerce, Linio y Facebook, de modo que si se realizaba un cambio en algún canal, este se realizaría en el resto de los canales del cliente.

Además de estos servicios, Astroselling ofrece la sincronización de órdenes con los distintos canales y un módulo de respuestas automáticas en base a ciertos criterios, utilizando inteligencia artificial. A modo de ejemplo, si un posible comprador hace una pregunta sobre stock de un artículo, Astroselling se fija si el artículo tiene stock y le responde automáticamente, sin que tenga que haber interacción humana.

A falta de un módulo de estadística y análisis es que surge la necesidad de iniciar este proyecto.

1.3. Objetivos

A continuación se describen los distintos objetivos planteados al inicio del proyecto. Estos determinaron la primera orientación del equipo para comenzar con el desarrollo inicial.

1.3.1. Objetivos académicos

Como alumnos universitarios el equipo desea aprender a manejar y gestionar correctamente un proyecto de desarrollo de Software de una duración más larga de la que se encontraba habituado.

Es importante para el equipo gestionar un proyecto desde el principio, definiendo los requerimientos, la arquitectura y la metodología de trabajo.

Se desea demostrar que el equipo está preparado para ingresar al mundo extraacadémico, aplicando los conocimientos obtenidos durante la carrera y validados en el proyecto actualmente presentado.

1.3.2. Objetivos del proyecto

El objetivo del proyecto consta en presentar un prototipo de una solución capaz de dar un valor agregado a Astroselling en lo que se refiere al procesamiento de datos, otorgando a sus usuarios estadísticas e información con la cual serán capaces de tomar decisiones que impacten positivamente en su negocio, dándoles una ventaja estratégica con respecto a su competencia.

Este prototipo ideado a lo largo de este proyecto también puede considerarse un "*Minimum Viable Product*" (MVP). Nuestro objetivo en concreto es que los clientes de Astroselling puedan acceder a este prototipo y obtener valor de él, para luego en etapas posteriores poder seguir construyendo sobre este.

1.3.3. Objetivos del cliente

El cliente del desarrollo, Astroselling, posee como objetivo del proyecto obtener un prototipo funcional con las funcionalidades descritas en la sección anterior "Objetivos del proyecto", de manera de poder realizar un una prueba a un público cerrado. A

partir de esta prueba el cliente decidirá si conservar el prototipo y desarrollarlo aún más con su propio equipo de desarrollo, el cual puede o puede no incluir a los alumnos involucrados en el desarrollo inicial del mismo.

2. Planteamiento del problema

2.1. Contexto del problema

Astroselling ha recabado a partir de reuniones con sus clientes, que varios de ellos desean poseer información condensada y fácil de visualizar acerca de las publicaciones y ventas por categorías dentro de Mercado Libre que les facilite la toma de decisiones estratégicas con respecto a su negocio.

A partir de esto surge la necesidad de crear un módulo de analítica que le brinde un valor agregado a sus clientes, a pedido de Astroselling.

En una primera instancia la solución está enfocada al público de Mercado Libre Uruguay, dado que es el área de operaciones principal de Astroselling al momento de empezar el proyecto.

2.2. Usuarios y necesidades

Con el objetivo de definir de manera más concreta al usuario objetivo y así obtener una validación previa de los requerimientos funcionales, el equipo decidió emplear la técnica de *"User Personas*". [3] Esta técnica consta de crear usuarios semi ficticios a partir de información recabada de entrevistas y preguntas al cliente y los usuarios objetivo. Estos usuarios semi ficticios intentan englobar características generales de los usuarios consultados.

Para modelar las personas se brindaron las siguientes preguntas a Astroselling, con la intención de que fueran distribuidas a sus clientes e interesados.

- ¿Es usted un usuario de Astroselling actualmente?
- ¿En caso de serlo, para que lo utiliza?
- ¿A que se dedica?
- ¿Siente que podría tomar decisiones de negocio con más seguridad si contara con información a su disposición?

A partir de las charlas con Astroselling donde se realizó una puesta en escena de las respuestas de los varios usuarios, se han podido identificar varios contextos de uso y

tipos usuarios para la aplicación de Astrolytics. Estos contextos fueron concretados con el uso del concepto de *User Persona* a continuación.

2.2.1. *User Personas*

Estas son las *User Personas* que fueron identificadas como potenciales usuarios de la aplicación. Astrolytics fue desarrollado tomando en cuenta estas *User Personas* como usuarios objetivo, influyendo en decisiones de diseño que requerían tomar en cuenta el público objetivo.

	¿Quiénes son?	¿Cuál es su objetivo?	¿Qué les impide alcanzar este objetivo?
Persona Nuevo Ingreso	Dueño de PyMEs o de un negocio pequeño con manufactura	Ingresar al mercado online con su actual producto o con una nueva inversión	No poseen la seguridad de que pueda ser remunerable invertir en la infraestructura para comenzar a distribuir de manera online
Persona Vendedor Establecido	Gerente de marketing o ventas de una empresa establecida	Mejorar las ventas de sus productos principales o agrandar su abanico de productos ofrecidos	No cuentan con información del mercado suficiente como para realizar movimientos que podrían resultar en pérdidas

Tabla 2.2.1.1. *User personas*

2.3. Requerimientos funcionales

Los requerimientos funcionales han sido agrupados en categorías para su fácil categorización.

Grupo	Id	Requerimiento
Gestión	RF1	Iniciar sesión con credenciales de Astroselling
	RF2	Cerrar Sesión
	RF3	Rotar tiendas de Mercado Libre
Explorar Mercado Libre	RF4	Explorar categorías de Mercado Libre
	RF5	Ver tendencias por categoría padre
	RF6	Ver cantidad de publicaciones por categoria
	RF7	Ver un desglose de ingresos por categoría
	RF8	Ver un desglose de dominio de vendedores por categoría
	RF9	Ver artículos más vendidos para una categoría
	RF10	Ver vendedores con mayores ganancias en una categoría
	RF11	Realizar una exportación de la información de categoría a Excel
	RF12	Marcar vendedores y publicaciones como favoritos
	RF13	Remover publicaciones y vendedores de favoritos
Favoritos	RF14	Seguimiento de stock para publicaciones favoritas
	RF15	Seguimiento de precio para publicaciones favoritas
	RF16	Seguimiento de *revenue* para tiendas

		favoritas
Búsquedas	RF17	Búsqueda de vendedores
	RF18	Búsqueda de publicaciones
Rankings	RF19	*Ranking* con el *top 100* de vendedores según *site*.
Vendedores	RF20	Perfil de vendedor con información detallada sobre ingresos y publicaciones
	RF21	Desglose de publicaciones por vendedor
	RF22	Realizar una exportación a Excel del perfil de un vendedor
	RF23	Comparar mi tienda con la de un vendedor
Métricas	RF24	Proporcionar una métrica para saber que tan saturada esta una categoría de Mercado Libre
	RF25	Proporcionar una métrica sobre el desempeño de una categoría y con respecto a sus hermanos
	RF26	Proporcionar una métrica respecto al *revenue* de una categoría contra el promedio total del vendedor
	RF27	Proporcionar una métrica respecto al *revenue* de una categoría contra las categorías en las que ya opera el vendedor
Feedback	RF28	Enviar *feedback* al equipo de desarrollo

Tabla 2.3.1. Requerimientos funcionales

2.4. Requerimientos no funcionales

Atributo de Calidad	Id	Requerimiento
Interoperabilidad	RNF2	Cumplimiento de estandares de comunicacion de sistemas (REST) [4][5]
Modificabilidad	RNF3	Facilidad de extensión
	RNF4	Interfaz de usuario desacoplada
Performance	RNF5	Tiempos de respuesta accesibles
Seguridad	RNF6	Autenticación
	RNF7	Confidencialidad
	RNF8	Uso de HTTPS
Testeabilidad	RNF9	Automatización de pruebas
Usabilidad	RNF10	Interfaz intuitiva

Tabla 2.4.0.1. Requerimientos no funcionales

2.4.1. Restricciones

Id	Restricción
Restricción 1	El código debe ser escrito en inglés y seguir los estándares de codificación del lenguaje usado

Tabla 2.4.1.1. Restricciones

2.5. Atributos de calidad

A partir de los requerimientos y contexto analizado, el equipo de desarrollo decidió enfatizar los siguientes atributos de calidad [6]:

2.5.1. *Performance*

Se trabajará con grandes volúmenes de datos, por lo que fue considerado importante tener la correcta *performance* del sistema en mente desde un comienzo, para evitar tiempos de carga y respuesta no aceptables para los usuarios.

La cantidad de usuarios que se maneja para el prototipo pensado para este proyecto es de 10, por lo cual en el peor de los casos se tendrán 10 conexiones concurrentes, lo cual nos permite tener un margen significativo para poder cumplir con la *performance*. Sin embargo, se tomarán los recaudos pertinentes para que la solución sea fácilmente escalable en cuanto a número de usuarios sin impactar en la *performance*.

2.5.2. Modificabilidad

Como el producto final fue acordado a ser cedido a Astroselling para su posterior desarrollo, se consideró importante que el sistema fuese fácilmente modificable, con el objetivo de reducir lo más posible el impacto de la cesión de derechos. Astroselling deberá capacitar y familiarizar a sus desarrolladores con el ambiente de desarrollo y la arquitectura de la aplicación entregada, por lo tanto el equipo asumió la responsabilidad de hacer esa transición lo más amena posible.

2.5.3. Interoperabilidad

Astrolytics se encontrará interactuando con los sistemas ya existentes de Astroselling y Mercado Libre, por lo que es óptimo para el sistema que se implementen estándares de comunicación de sistemas, con tal de evitar problemas de comunicación.

2.5.4. Usabilidad

Dado que el producto será utilizado por usuarios que no necesariamente son familiares con sistemas informáticos, este deberá ser fácil de utilizar e intuitivo de aprender.

3. Descripción de la solución

La solución ideada consiste de un sistema separado en módulos, los cuales se comunican a través de llamadas HTTP y APIs definidas. El sistema se encarga de obtener información de Mercado Libre, para luego procesarla en métricas y almacenarlas para su consumo posterior. Estas métricas pueden ser accedidas por el usuario desde el módulo que actúa como *frontend*, Commander.

3.1. Descripción funcional

Astrolytics presenta un portal de autenticación al entrar a la aplicación, el cual utiliza las credenciales de Astroselling.

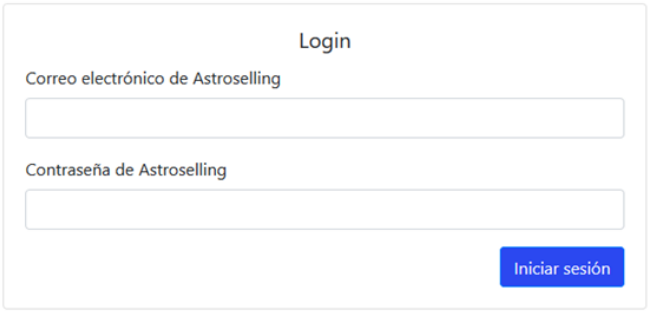


Imagen 3.1.0.1. *Login*

Una vez dentro, el usuario es recibido por un *dashboard* presentando información relevante al usuario. Esta información se adecua más y más conforme la aplicación es usada, gracias a las opciones de customización ofrecidas. También puede encontrarse en la barra de navegación un *dropdown* el cual permite seleccionar entre las distintas tiendas de Mercado Libre que el usuario posee.

Actualmente, la selección de tienda afecta las secciones de recomendaciones y comparaciones, dado que estas utilizan información específica y relevante a la tienda del usuario.

Al costado se encuentra un menú que presenta las principales funcionalidades de la aplicación, mostrable clickeando el botón arriba a la izquierda.

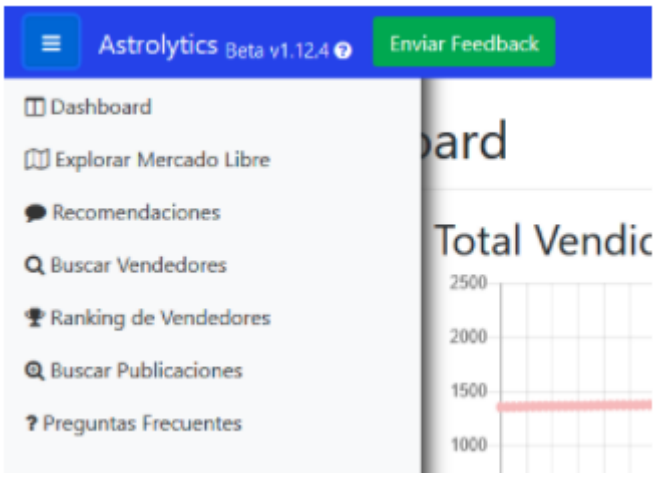


Imagen 3.1.0.2. *Sidebar* de Astrolytics

3.1.1. *Dashboard*

El *dashboard* permite al usuario tener un seguimiento resumido de sus publicaciones favoritas. Entre los datos presentados se puede encontrar el seguimiento diario del stock disponible de los productos, la cantidad vendida y su precio en dólares o pesos uruguayos.

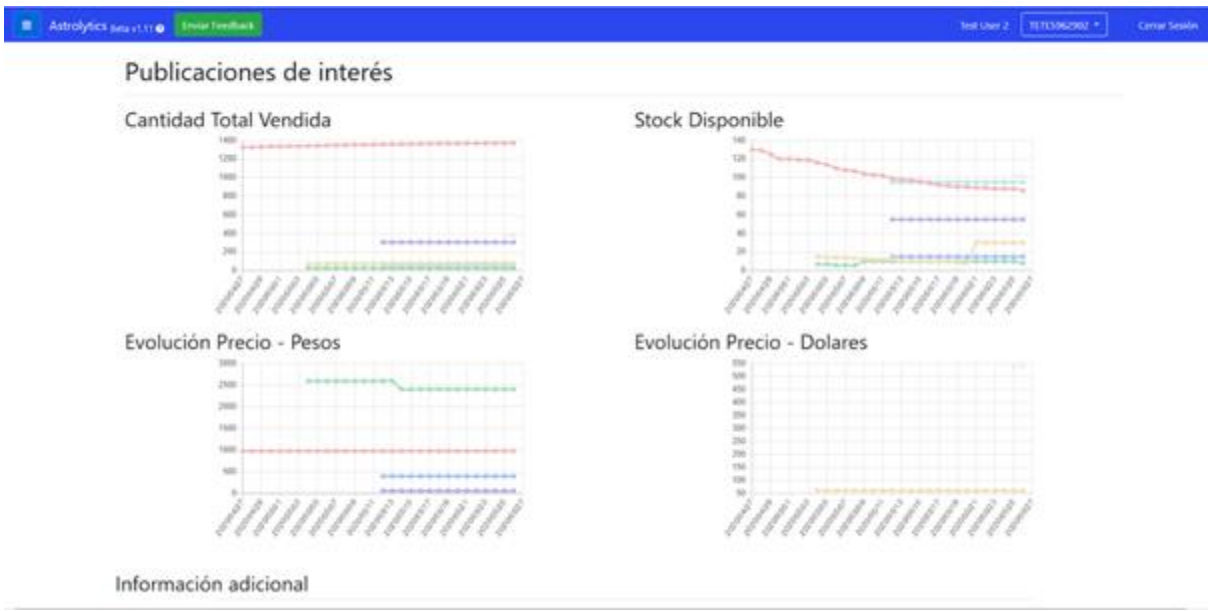


Imagen 3.1.1.1. *Dashboard* de Astrolytics

También es posible encontrar una gráfica del *revenue* diario para las tiendas marcadas como favoritas.

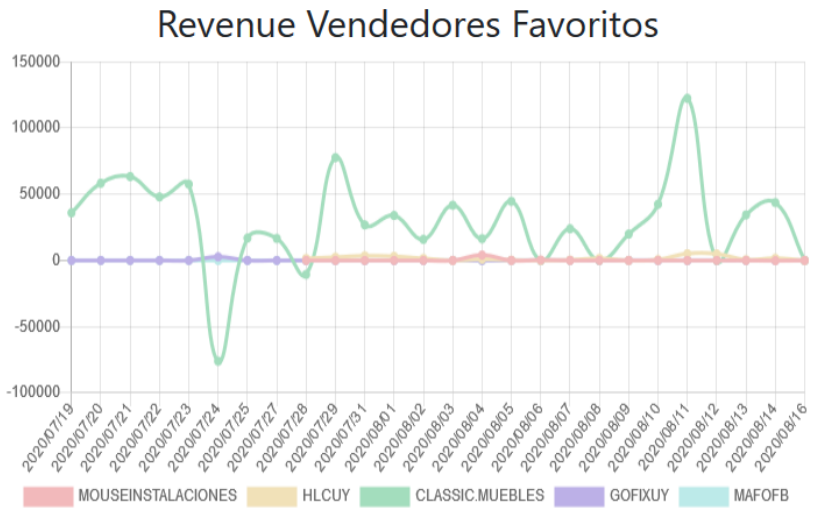


Imagen 3.1.1.2. Gráfica de *revenue* de vendedores en el *dashboard*

Las publicaciones favoritas pueden ser administradas en la sección de “Explorar Mercado Libre”, el *dashboard* también presenta la posibilidad de eliminar favoritos de manera rápida.

Color	ID	Título	Var. Stock 24hs	Var. Stock 7d	Eliminar
	MLU446982869	Ps3 160gb Ref. + 10 Juegos (fifa 19 Gta5 Etc) + 2 Controles	-5	-25	✖
	MLU444602054	Tobi Pesas 2kg Entrenamiento Piernas Tobillera Velcro El Rey	-3	245	✖
	MLU450609170	Bolsos Maternales / Mochila Para Bebés - Todos Los Colores	0	56	✖
	MLU445226255	Baúl Plástico Delivery 80lts Blanco O Negro Reforzado	0	0	✖
	MLU470675359	Tapaboca Barbijo Mascarilla Tnt Lavable Ecologico 2 Unidades	0	0	✖
	MLU470823190	Tapa Bocas Tela Lavables, Reutilizables	0	0	✖
	MLU443895394	Colchoneta 100x70 Espesor 5cm Gimnasia Abdominal Yoga El Rey	0	408	✖
	MLU451323607	Auto Jeep Compact Batería C.remot Luz Música Español El Rey	0	132	✖
	MLU470676048	Banda Suspensión Entrenamiento Profesional Expert El Rey	0	161	✖
	MLU448086314	Baul Delivery Para Moto 80 Lts Protork Negro O Blanco	N/A	N/A	✖
	MLU471167914	Protector Facial Mascarilla Acetato Tapaboca	N/A	N/A	✖
	MLU445198295	Ps4 Playstation 4 1tb Nueva+ 110 Freegames+fortnite+apex	N/A	N/A	✖

Imagen 3.1.1.3. Administración de favoritos

3.1.2. Explorar Mercado Libre

La exploración permite al usuario moverse a través del árbol de categorías que posee Mercado Libre, pudiendo discernir cuales son las categorías más populares a partir de la cantidad de publicaciones que estas poseen. Las categorías más básicas, que no poseen ramificación alguna son denominadas como categorías hoja.

Imagen 3.1.2.1. Sección de exploración

Uno puede consultar las últimas tendencias de productos en las categorías de más alto nivel, haciendo click en el ícono de gráficas en amarillo de esta sección.

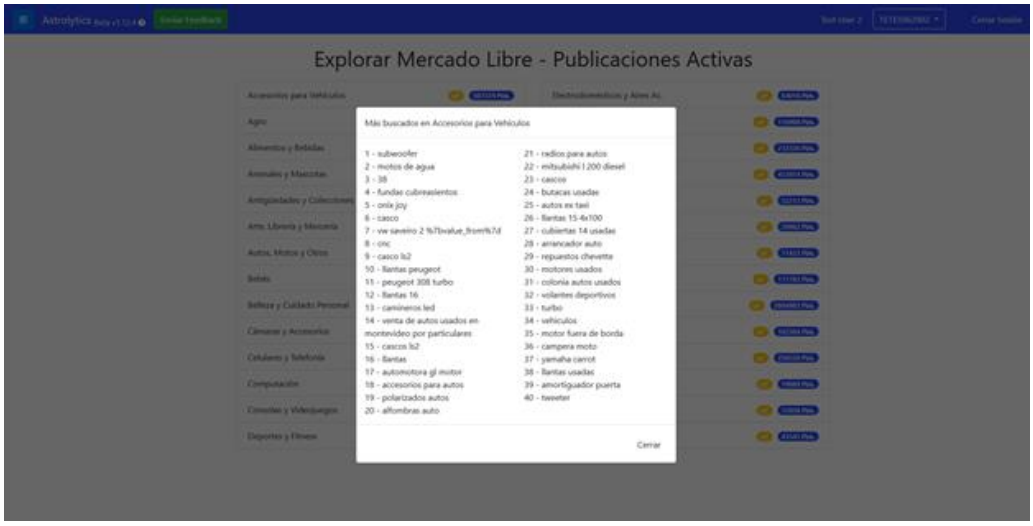


Imagen 3.1.2.2 - Tendencias de categorías

Dentro de las categorías hoja, se presentan gráficas desglosando la distribución de ingresos entre vendedores, junto con un *ranking* de las publicaciones y vendedores más exitosos de la categoría. En este *ranking* también es posible marcar publicaciones y vendedores como favoritos para tener un seguimiento de ellos en el *dashboard.*

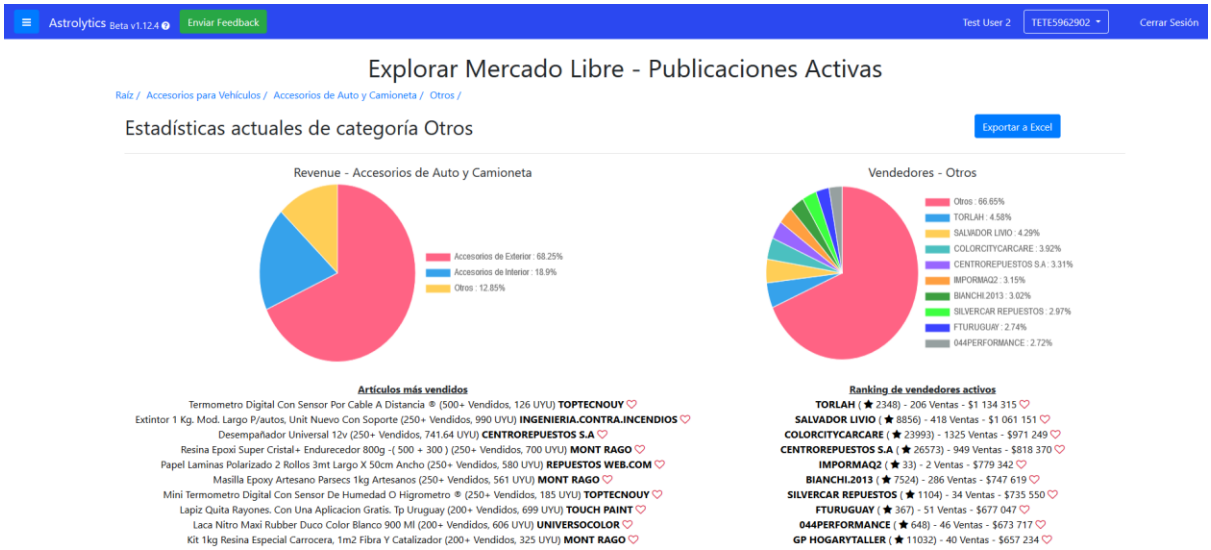


Imagen 3.1.2.3 - Vista de categoria hoja

El *ranking* de artículos presenta para cada uno su nombre, una aproximación de sus cantidades vendidas obtenida de la API de Mercado Libre, el *revenue* total de la publicación en base a las cantidades vendidas, y su vendedor. Juntos con estos datos también se brinda un *link* a la página de Mercado Libre del artículo, así también un *link* directo al perfil del vendedor, descrito en el punto 3.1.7.

Con respecto al *ranking* de vendedores se puede encontrar listado para cada vendedor su nombre junto con la cantidad de *ratings* que ha recibido en Mercado Libre, sus ventas y su *revenue* total. Cada punto provee también un *link* directo al perfil del vendedor.

En ambos casos, para cada punto de los *rankings* es posible marcarlos como favoritos con un click en el corazón.

Los precios de las publicaciones, junto con los *revenues* listados, son convertidos a pesos uruguayos en caso de encontrarse en dólares originalmente, calculado con la

cotización diaria. Esto se realiza con el objetivo de permitir una comparación monetaria de manera más fluida.

3.1.3. Recomendaciones

En la sección de recomendaciones es posible encontrar una tabla de categorías relacionadas a las que el usuario actualmente opera. En esta tabla se pueden encontrar 4 métricas calculadas y asignadas a cada categoría.

Imagen 3.1.3.1. Recomendaciones de categorías

Actualmente se definen y ofrecen cuatro métricas:

- ***Mean Category Revenue Index:*** Cómo se maneja la categoría con respecto al *revenue* promedio total de las categorías que tiene el vendedor. Mientras más alto, mejor.
- ***Mean Seller Revenue Index:*** Cómo se maneja el *revenue* de la categoría con respecto al *revenue* promedio total del vendedor. Mientras más alto, mejor.
- ***Brother Revenue Index:*** Que tan bien está el *revenue* de la Categoría con respecto al *revenue* promedio de sus hermanos. Mientras más alto, mejor.
- ***Saturation Index:*** Mientras más alto, más saturada la categoría.

Las métricas también se marcan en colores, los cuales apuntan a indicar cuáles categorías son las más favorables para que el usuario pruebe operar.

3.1.4. Buscar Vendedores

La búsqueda de vendedores permite realizar una búsqueda rápida de los usuarios de Mercado Libre, permitiendo ingresar rápidamente a su perfil en esta plataforma o a su página de vendedor en Astrolytics. Los vendedores se muestran en orden de importancia, tomando como criterio el tipo de vendedor y las calificaciones obtenidas por sus ventas.

Imagen 3.1.4.1. Sección de búsqueda de vendedores

3.1.5. Búsqueda de Publicaciones

La búsqueda de publicaciones permite realizar una búsqueda de publicaciones de Mercado Libre, listando su stock, precio y permitiendo al usuario marcar publicaciones como favoritas.

Imagen 3.1.5.1. Sección de búsqueda de publicaciones

Cabe mencionar que la búsqueda realizada consume la API de Mercado Libre, por lo tanto los resultados de la búsqueda en Astrolytics son equivalentes a la misma búsqueda realizada en el sitio de Mercado Libre. También se cuenta con un paginado para los resultados, dado que se puede obtener hasta 50 páginas de resultados por búsqueda.

3.1.6. Ranking de Vendedores

El *ranking* de vendedores presenta una vista ordenada de los vendedores más exitosos de Uruguay. Esto se calcula con una comparación de los ingresos históricos totales.

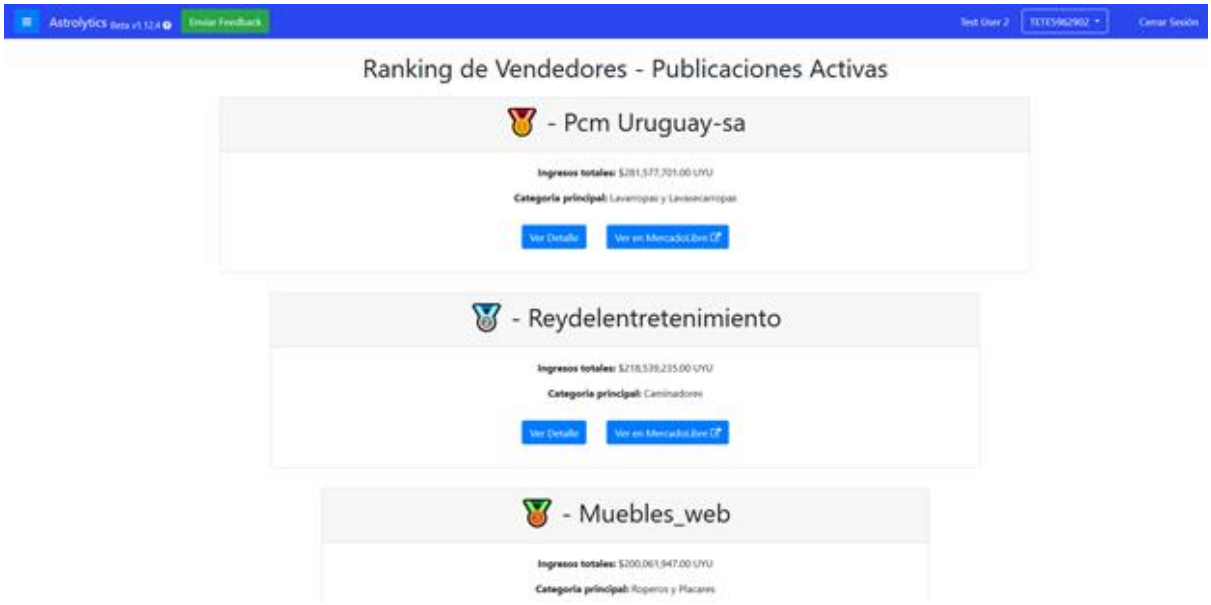


Imagen 3.1.6.1. Sección de ranking de vendedores

A partir de cada una de las entradas, uno puede acceder a su perfil de Astrolytics o de Mercado Libre. También se puede encontrar en cada una el valor del ingreso histórico total para el vendedor utilizado para el posicionamiento del *ranking*, junto con la categoría más próspera del vendedor.

El *ranking* es actualizado diariamente con los datos al día de los usuarios.

3.1.7. Perfil del Vendedor

En el perfil de un vendedor es posible encontrar varios datos relacionados al desempeño de un vendedor de Mercado Libre.

La intención de esta sección es brindarle al usuario la posibilidad de investigar otros usuarios de interés, permitiéndoles realizar un análisis de negocio detallado y así poder sacar conclusiones, de manera que puedan ayudar a fomentar el negocio de sus propios emprendimientos.

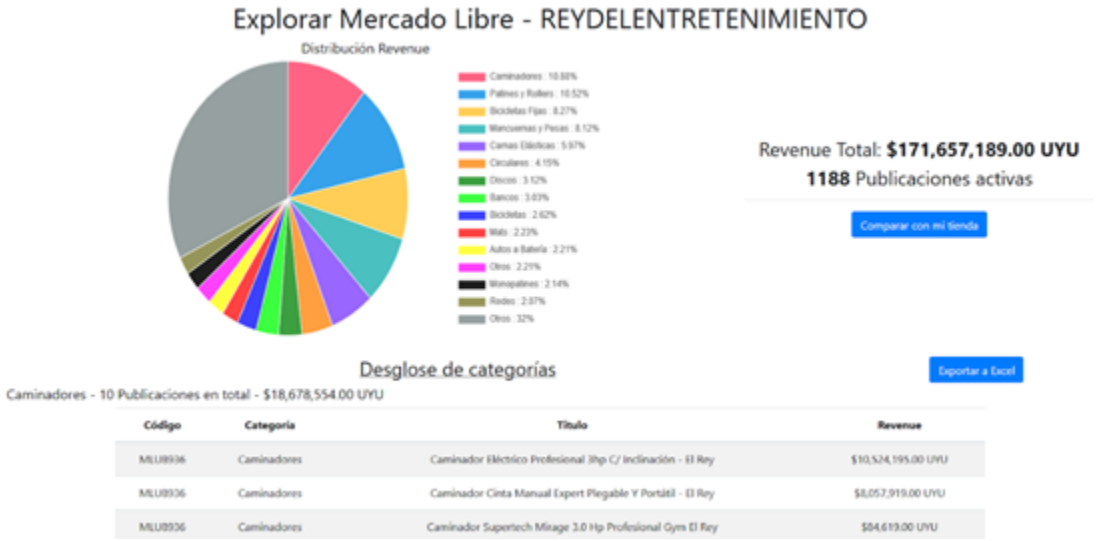


Imagen 3.1.7.1. Perfil de un vendedor

Por un lado se presenta una gráfica de distribución de ingresos del vendedor agrupados por categoría. Esta muestra el porcentaje de ingresos que cada categoría activa del vendedor aporta a su total de ingresos históricos. Solamente se muestran las 14 categorías más prolíferas del vendedor, con tal de mantener la legibilidad de la gráfica a un nivel aceptable. El resto de las categorías que son demasiado pequeñas son categorizadas en conjunto dentro de la etiqueta “Otros”.

Al costado de la gráfica de distribución se puede encontrar un resumen del *revenue* histórico total del vendedor, junto con el dato de sus publicaciones actualmente activas en Mercado Libre. También es posible acceder a la comparación de tiendas utilizando el botón de “Comparar con mi tienda”. Esto permite realizar una referencia cruzada de los datos de desempeño de la tienda perteneciente al perfil contra la del usuario ingresado actualmente.

Debajo de la gráfica se presenta el desglose de categorías. Aquí se puede encontrar tablas referentes a cada una de las categorías etiquetadas en la gráfica, brindando detalles de cada una de las publicaciones del vendedor en esa categoría. Los datos brindados para cada fila son código de publicación, categoría, título y revenue total del producto. Es posible exportar todos estos datos a un archivo Excel utilizando el botón “Exportar a Excel” posicionado sobre el desglose de categorías.

Es importante mencionar que el perfil de vendedor no está disponible para todos, únicamente para aquellos vendedores que hayan logrado superar el margen de diez mil pesos uruguayos en ventas de Mercado Libre. Esta decisión fue llevada a cabo en base a que debía realizarse un filtro para almacenar la información que sería utilizada para la creación de métricas, con tal de ahorrar espacio y brindar un estándar a los datos proporcionados por el sistema. En caso contrario, el sistema se llenaría de información que no sería relevante para los usuarios objetivos. Un ejemplo de esto sería información de usuarios que no poseen un negocio, sino que solo han vendido pertenencias viejas en Mercado Libre.

3.1.8. Comparación de tiendas

En la comparación de tiendas uno puede encontrar una vista punto a punto de la tienda propia y la seleccionada, donde se comparan varios elementos de las mismas. Entre estos puntos uno puede encontrar información como tiempo de registro, nivel de usuario, puntos de Mercado Libre, localidad aproximada, gráficas de transacciones y *ratings*, y finalmente un desglose de publicaciones y finanzas.

Imagen 3.1.8.1. Comparación de tiendas

3.2. Descripción técnica

A nivel técnico, Astrolytics está compuesto por un ecosistema de aplicaciones separadas por responsabilidades. También utiliza servicios periféricos, como una base de datos MongoDB y una carpeta de Google Drive dedicada a contener los datos brutos recabados. Cada una de estas aplicaciones o módulos posee un nombre interno de desarrollo, entre los cuales se puede encontrar Commander, Analytics, Scraper y Scheduler. A continuación se encuentran los detalles de cada uno.

3.2.1. Commander

El módulo commander conforma la interfaz de usuario principal. La responsabilidad del mismo es proveer al usuario la experiencia de la aplicación y presentar los datos pertinentes al usuario de manera fácilmente digerible. Commander releva todo procesamiento intenso y obtención de datos al *backend*, puntualmente el módulo Analytics.

3.2.2. Analytics

Analytics conforma la inteligencia principal del programa, se encarga de procesar la información obtenida y almacenada en la base de datos para su posterior consumo. Como consecuencia de esto, también ofrece una API REST para el consumo de estos datos. Por otro lado, proporciona funcionalidades como enviar mails cuando es necesario y proporcionar un *endpoint* de autenticación para el *frontend*.

3.2.3. Scraper

La responsabilidad de Scraper consiste en recabar información en masa de Mercado Libre y almacenarla para su posterior uso. Entre sus formas de obtener datos se puede encontrar una búsqueda completa de Mercado Libre, obtención de datos de un usuario en específico o de una publicación en concreto.

3.2.4. Scheduler

Scheduler tiene una responsabilidad muy puntual, que es la de orquestar las funcionalidades de procesamiento asincrónicas que se pueden encontrar en la

aplicación. Esto hace que se procesen y almacenen los datos para el consumo del *frontend* fuera de horas pico de uso, con tal de impactar lo menos posible la experiencia del usuario.

3.3. Vistas

A continuación se presentan varias vistas que apuntan a desglosar la solución desarrollada en sus distintas dimensiones. Estos vistas y diagramas están basados en las vistas descritas en “*Documenting Software Architectures: Views and Beyond*”, junto con el método asociado de presentación, utilizando una representación primaria, descripción de sus elementos y decisiones de diseño relevantes. [7]

3.3.1. Vista de *deploy*

3.3.1.1. Astrolytics - MVP

3.3.1.1.1. Representación Primaria

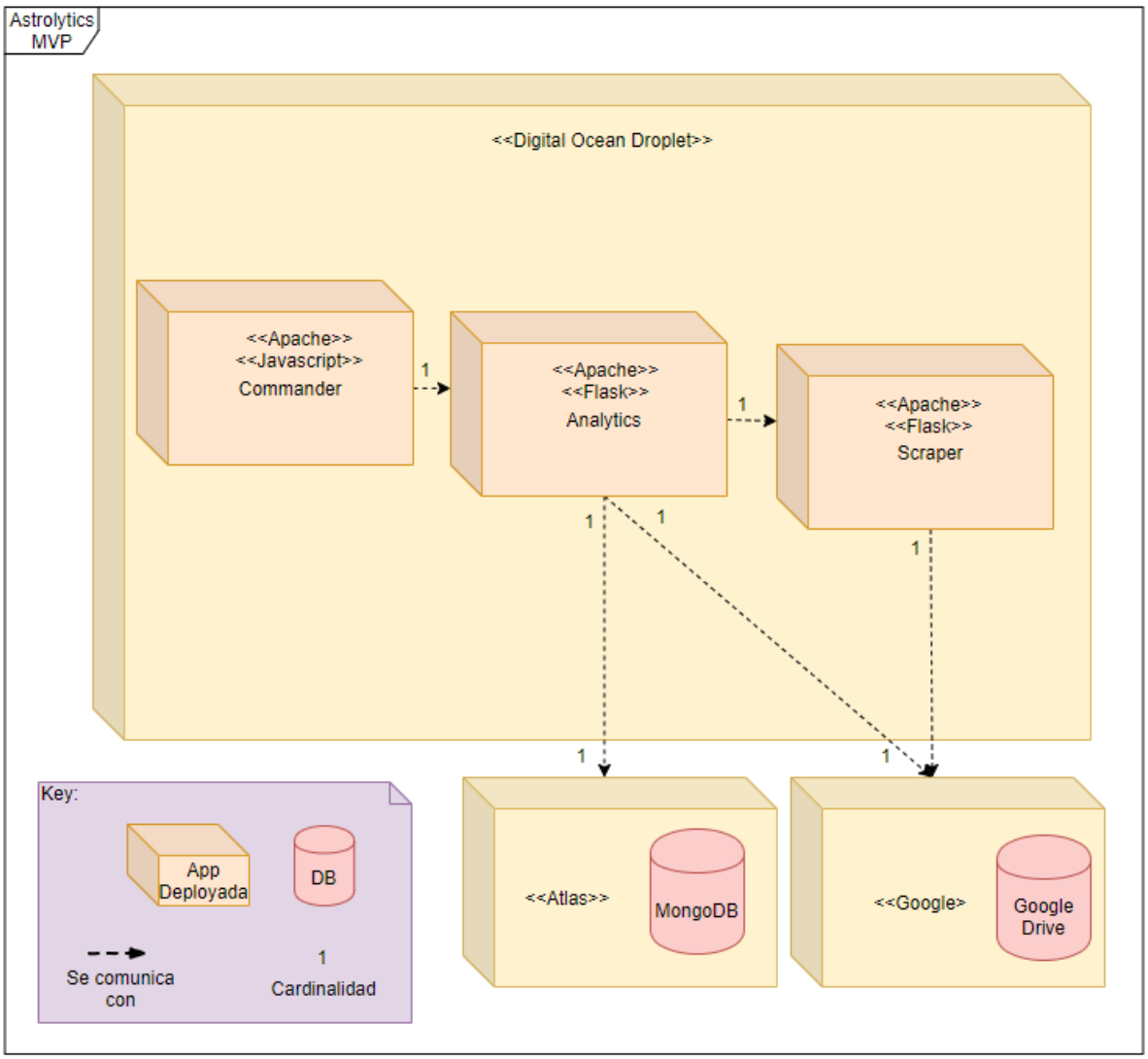


Imagen 3.3.1.1.1.1. Vista de *deploy* para Astrolytics

3.3.1.1.2. Catálogo de elementos

Nombre	Descripción
Digital Ocean Droplet	Digital Ocean es el proveedor de servicios para el *hosting* del servidor. El servidor consiste en una máquina virtual de Linux con Ubuntu, con 2 GB de RAM y 25 GB de SSD. Tiene una instalación de Apache donde están agregadas como sitios los tres componentes de Astrolytics.
Commander	Commander es una página con Javascript estático, que se ejecuta desde el navegador del usuario. La dirección para acceder a Commander es stats.astroselling.com
Analytics	Es una API deployada gracias a Flask y WSGI utilizando un entorno virtual de Python en el Apache. Para poder acceder a Analytics, se puede utilizar la url api.astrolytics.info.
Scraper	Tiene una estructura idéntica a Analytics en su *deployment*. Para poder comunicarse con Scraper, se puede utilizar la dirección scrapper.astrolytics.info
Atlas DB	Servicio de base de datos *as a service* que ofrece MongoDB. Hay una gran diferencia de velocidad en el ambiente de “producción” para el MVP que local, pudiendo realizar las consultas e inserciones mucho más rápido.
Google	Los servidores de Google proveen el servicio de almacenamiento de los archivos.
Scheduler	Scheduler no está como una figura como tal en el diagrama, ya que forma parte de un crontab con comandos cURL para ejecutar llamadas a URLS en determinados horarios, es algo propio del *droplet* aunque virtualmente sea un “componente”, no se considera como parte del proceso de *Deploy*.

Tabla 3.3.1.1.2.1. Catálogo de elementos de vista de *deploy*.

3.3.1.1.3. Decisiones de diseño

En una primera instancia, el MVP estaba hosteado en IBM gracias a un acuerdo que el cliente Astroselling poseía con el proveedor. En esos momentos, cada uno de los componentes era una *Web App* diferente y la base de datos estaba hosteada también en IBM. Cuando ese acuerdo se canceló inesperadamente, se tuvo que hacer una migración lo más rápida posible, fue entonces que se decidió usar el servicio de *droplets* de Digital Ocean.

Considerando que el objetivo principal de este proyecto es brindar un MVP, se cree conveniente realizar el *deploy* de la forma más práctica y económica posible, de manera que el servicio que se quiere brindar sea funcional. El primer desafío que se tuvo al realizar la migración a Digital Ocean fue el procesamiento de los datos y la cantidad de memoria necesaria para poder contener el caudal de archivos que se generan por Scraper, que generan aproximadamente 3 gigas de información al día sin comprimir. En un principio se optó por un servidor con menos memoria RAM, pero se tuvo que hacer un upgrade al *droplet* de 2 gigas para poder hacer que la plataforma funcione correctamente.

El caudal de datos que fue calculado para la base de datos terminó por descartar la idea de tener la base de datos también en la misma máquina virtual, por lo cual se optó por utilizar el servicio de Atlas que proporciona MongoDB. Como muchas cosas en este proyecto, el concepto que se tenía de la base de datos fue modificándose y ahora la información que se almacena es poca, por lo cual la base de datos actual de MongoDB que se tiene en Atlas funciona como una especie de caché para los datos de procesamiento diarios de Analytics, por lo cual, quizás hubiera sido más efectivo utilizar un grupo de servidores de caché de Redis en el Droplet de Digital Ocean, que es algo que se tendrá que evaluar para una segunda etapa del proyecto más allá del MVP.

En cuanto al almacenamiento de los datos del Scrapeo en el servicio de google, se trató de cumplir con la misma premisa: La capacidad que se tiene es solo de 25 GB, por lo cual eventualmente se llenaría por el backup de la información. Cada archivo comprimido por día agrega 150 megas al servidor, por lo cual no se consideraba viable a largo plazo con la capacidad que brinda Digital Ocean. Quizás más adelante se

podría utilizar un servidor de *hosting* dedicado o algún servicio de IBM o Azure de *data storage*, pero la solución que ofrece Google es gratis y funciona apropiadamente, por lo cual no se siente la necesidad de cambiarla a corto plazo.

3.3.2. Vista de componentes y conectores

3.3.2.1. Astrolytics

3.3.2.1.1. Representación Primaria

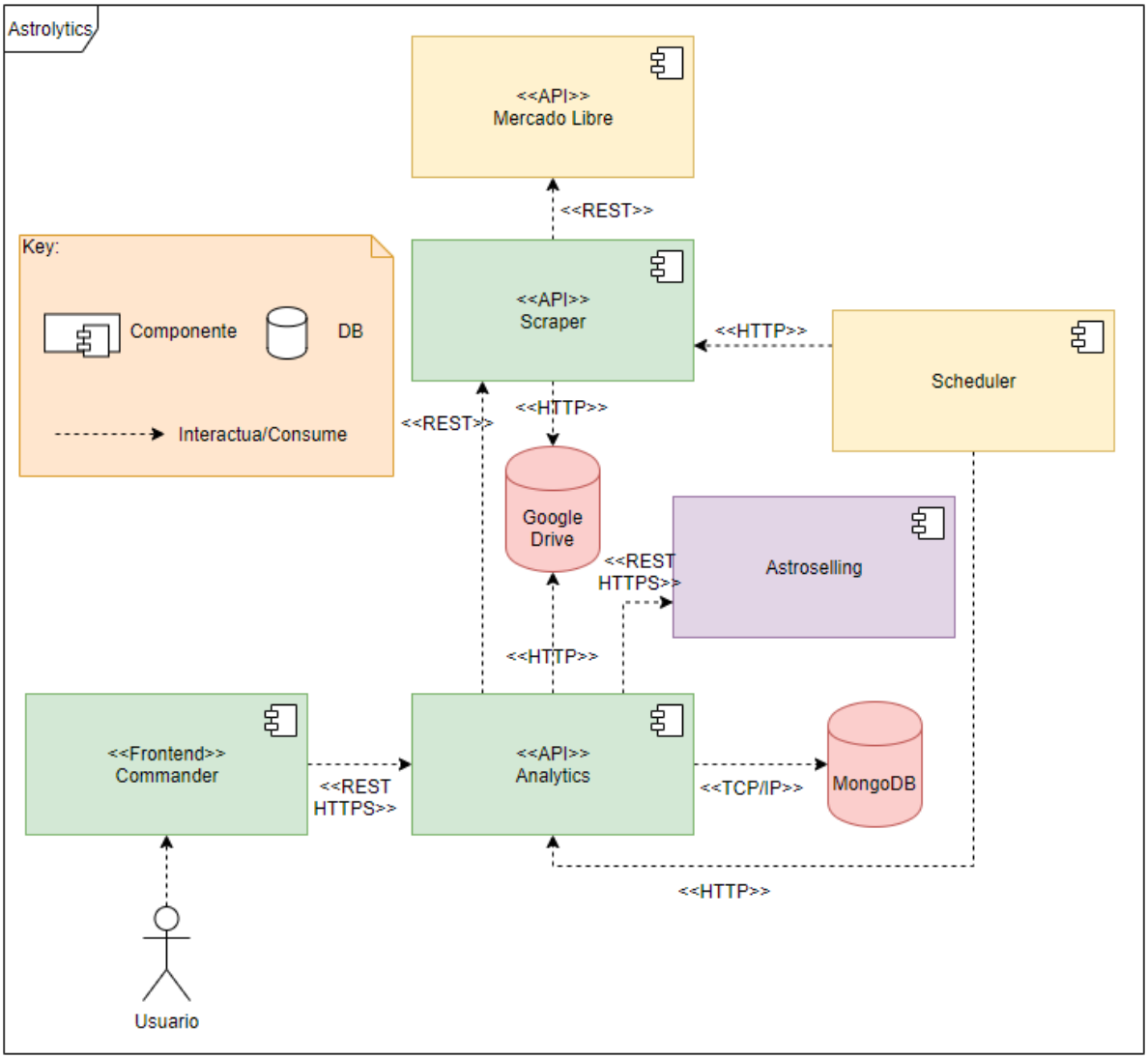


Imagen 3.3.2.1.1.1. Vista de componentes y conectores para Astrolytics

3.3.2.1.2. Catálogo de elementos

Componente	Tipo	Descripción
Mercado Libre	Plataforma Web	Mercado Libre, es accedida por Scraper vía API para obtener información de las publicaciones de las categorías, y vía HTML cuando se requiere obtener información más específica de una publicación que la API no puede proporcionar.
Scraper	Web *Service*	Servicio de Scrapeo de Astrolytics, obtiene la información de Mercado Libre y la devuelve a Analytics o la sube a Google Drive para su posterior procesamiento.
Scheduler	Crontab/cURL	Scheduler se encarga de hacer que Scraper y Analytics ejecuten sus procesos en lote todos los días, mediante llamadas HTTP mediante cURL a las APIs de estos a ciertas horas del día, definidas en un *crontab*.
Google Drive	Servicio de Almacenamiento	Servicio de almacenamiento de Google para subir la información que se scrapea de Mercado Libre. La información es depositada por Scraper y leída por Analytics.
Astroselling	Plataforma Web	Astroselling es la plataforma del cliente. Analytics la usa para validar el login de los usuarios que tratan de entrar a Astrolytics.
Analytics	Web Service	API Rest que se encarga de toda la lógica de procesamiento de la información, y le brinda al componente Commander toda la información que este le pide.
MongoDB	Base de datos no relacional	Donde se guarda toda la información procesada por Analytics.
Commander	Aplicación Web	*Frontend* de la aplicación, el usuario interactúa con él y él interactúa exclusivamente con Analytics.

Tabla 3.3.2.1.2.1. Catálogo de elementos de vista de componentes de Astrolytics.

3.3.2.1.3. Decisiones de diseño

Las decisiones de diseño más importantes de esta sección abarcan la definición de arquitectura principal del proyecto, por lo cual se explican con lujo de detalles en el apartado Diseño de la Arquitectura dentro del Proceso de Desarrollo.

3.3.3. Vista de módulos

3.3.3.1. Scraper - Vista de Descomposición

3.3.3.1.1. Representación Primaria

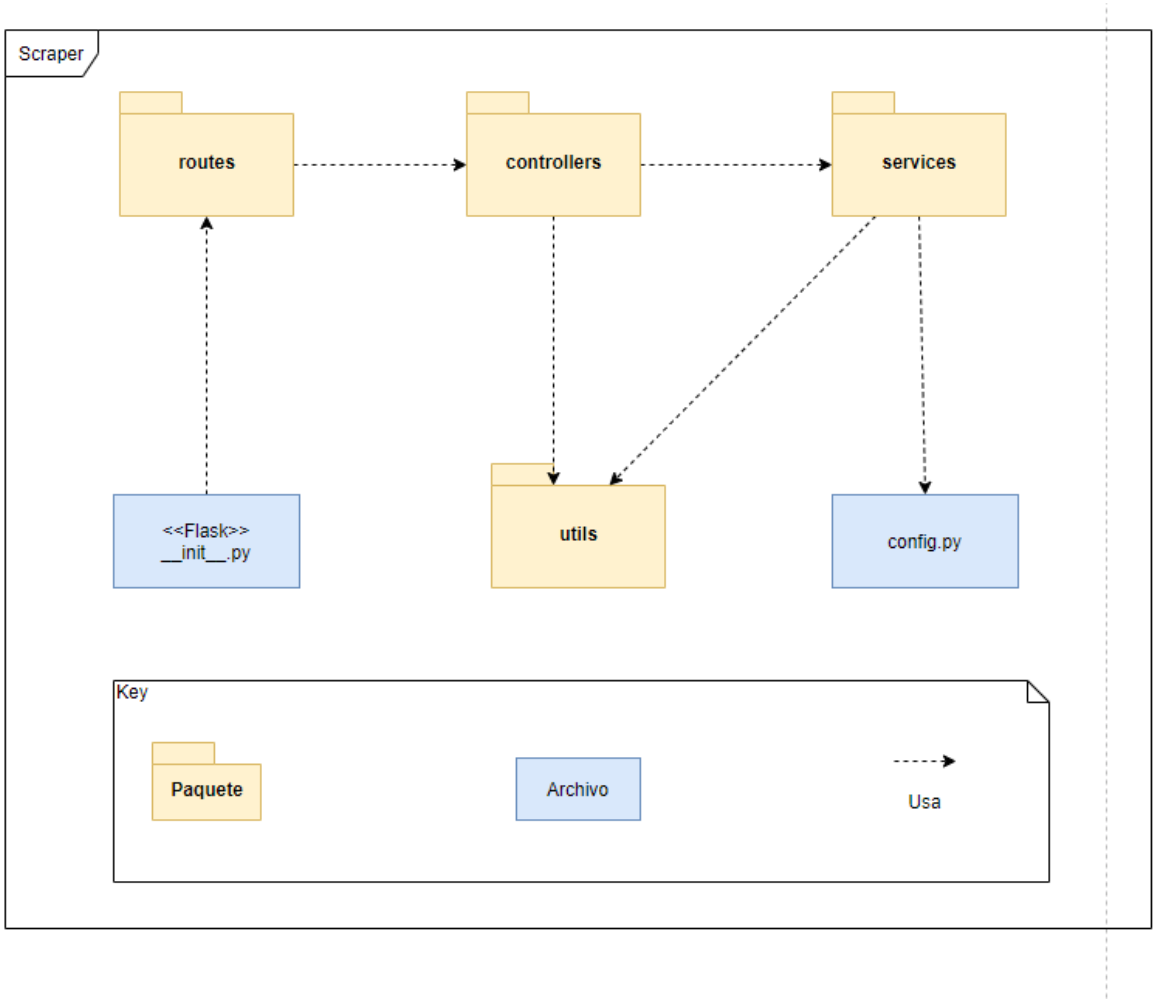


Imagen 3.3.3.1.1.1. - Vista de módulos para Scraper

3.3.3.1.2. Catálogo de elementos

Elemento	Responsabilidades
__init__.py	Es el punto de entrada de la aplicación, es el archivo encargado de ejecutar el *framework* de Flask con la configuración dada por el entorno.
routes	Es la carpeta donde están los archivos que declaran las rutas y las llamadas a los controladores.
controllers	La carpeta que define a los controladores, son el punto medio entre las rutas y la capa de servicios. El único controlador en esta carpeta es ScrapController.
services	En este paquete están las clases de servicio, encargados de realizar la lógica de negocio. Los archivos contenidos aquí son ListingServices, MeliServices, ScrapServices y StorageServices. Los primeros dos son para manejar lógica de Mercado Libre, el tercero es para manejar la lógica del Scrapeo, y el último es para poder comprimir y guardar los archivos en el Google Drive
utils	Paquete que define algunas funciones auxiliares para el I/O de archivos, como por ejemplo borrar archivos, carpetas, archivos de lock, listar archivos, cargar jsons.
config.py	Archivo donde están definidas las variables de configuración, obtenidas generalmente de las variables de entorno o en su defecto, de valores predeterminados en el archivo.

Tabla 3.3.3.1.2.1. Catálogo de elementos de vista de módulos de Scraper.

3.3.3.1.3. Decisiones de Diseño

Este componente de la aplicación tiene la responsabilidad única de manejar el scrapeo de las publicaciones en Mercado Libre. Por lo tanto, es bastante simple en cuanto a diseño. Se quiso imitar el flujo de una aplicación web enfocada a ser una API REST con flujo de ruta controlador servicio que maneja la lógica de negocio. El archivo de configuración es accedido por la capa de servicios para algunas configuraciones como credenciales del Google Drive, o credenciales para poder utilizar la API de Mercado Libre, así como los valores predeterminados de las rutas de las carpetas para guardar los resultados de Scrapeo.

Hay algunos detalles que fueron omitidos en la representación primaria porque son archivos donde se guarda información para procesar algunos datos. Por ejemplo, el árbol de categorías de Mercado Libre, que es actualizado una vez a la semana mediante consultas a la API. Esto permite buscar las publicaciones por categoría de una manera mucho más eficiente.

3.3.3.2. Analytics - Vista de Descomposición

3.3.3.2.1. Representación Primaria

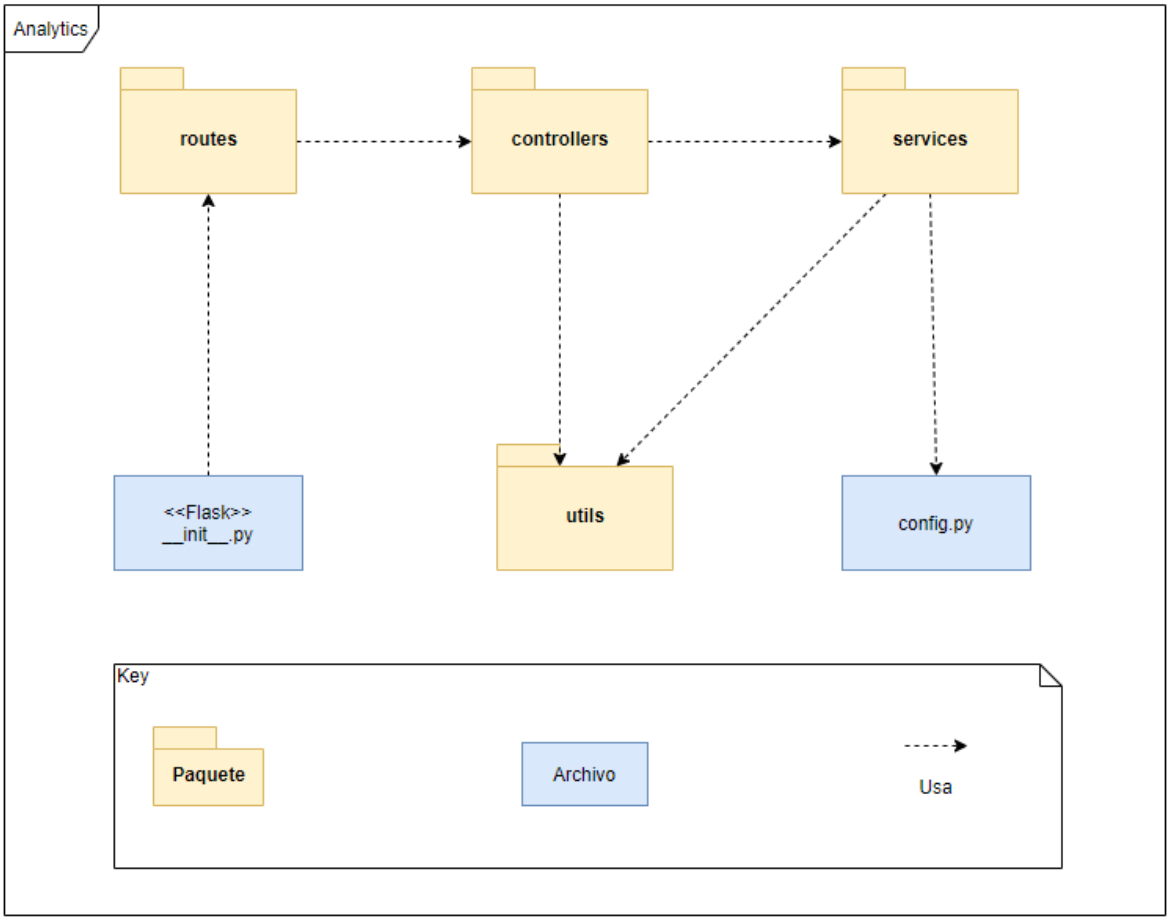


Imagen 3.3.3.2.1.1. Vista de módulos para Analytics

3.3.3.2.2. Catálogo de elementos

Elemento	Responsabilidades
__init__.py	Es el punto de entrada de la aplicación, es el archivo encargado de ejecutar el *framework* de Flask con la configuración dada por el entorno.
routes	Es la carpeta donde están los archivos que declaran las rutas y las llamadas a los controladores.
controllers	La carpeta que define a los controladores, son el punto medio entre las rutas y la capa de servicios. Los controladores que existen son AnalyticsController (utilizado para la analítica en general), AstrosellingController (utilizado para manejar el *login* con Astroselling), FavoritesController (punto de entrada para operaciones referentes a favoritos), MercadoLibreController (consultas de Mercado Libre), TrendsController (consultas referentes a tendencias), entre otros.
services	En este paquete están las clases de servicio, encargados de realizar la lógica de negocio. Algunos archivos que se pueden encontrar aquí son AnalyticsServices, AstrosellingServices, ExchangeServices (para manejar la obtención de la cotización del dólar y otras monedas), FavoriteServices, GoogleDriveServices (para obtener los archivos de GoogleDrive), MercadoLibreServices, MongoServices (Manejo de operaciones para la base de datos), TrendServices, entre otros.
utils	Paquete que define algunas funciones auxiliares para el I/O de archivos, como por ejemplo borrar archivos, carpetas, archivos de lock, listar archivos, cargar jsons.
config.py	Archivo donde están definidas las variables de configuración, obtenidas generalmente de las variables de entorno o en su defecto, de valores predeterminados en el archivo.

Tabla 3.3.3.2.2.1. Catálogo de elementos de vista de descomposición Analytics

3.3.3.2.3. Decisiones de Diseño

Si se observa con atención, se podrá descubrir que la representación primaria de esta vista es igual al del sistema Scraper. Esto es porque sirve el mismo propósito que Scraper, pero a otra escala (es decir, es una API REST hecha en Python), y porque se utilizó como template la primera para realizar Analytics. Por lo tanto, a pesar de que el contenido de los paquetes cambien, contienen la misma estructura y organización.

3.3.3.3. Commander - Vista de Descomposición

3.3.3.3.1. Representación Primaria

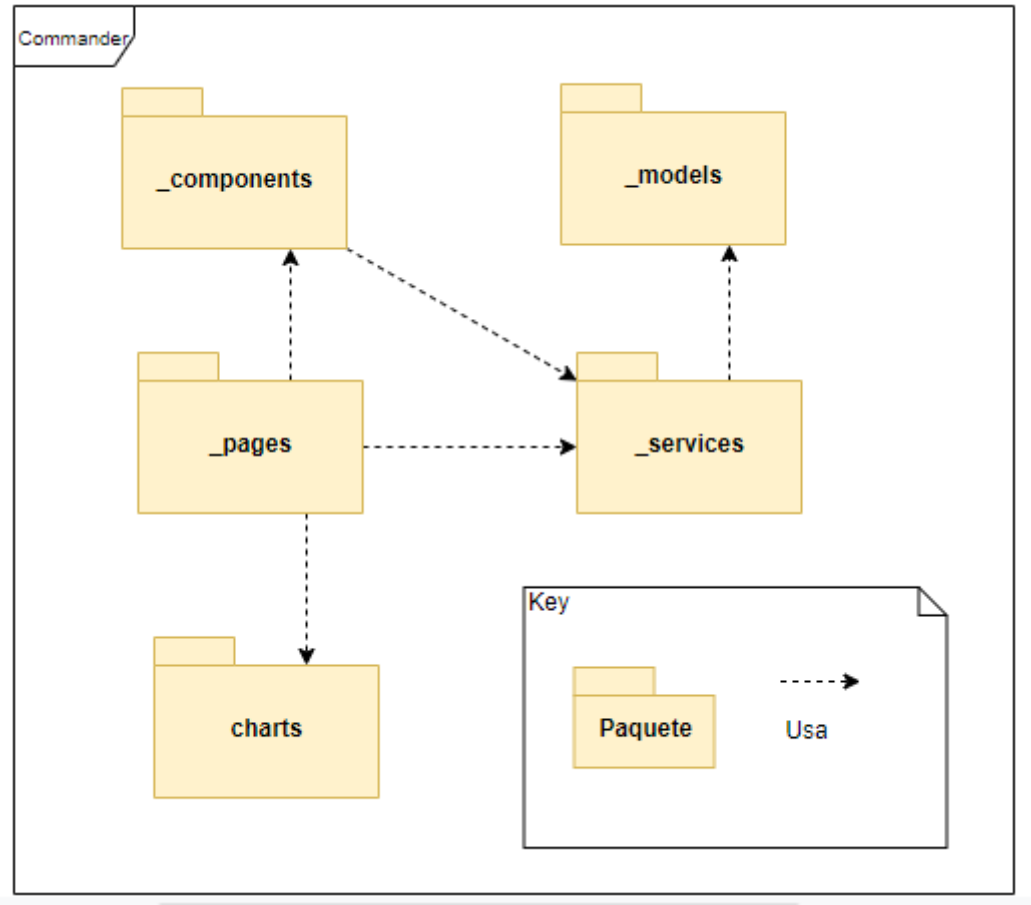


Imagen 3.3.3.3.1.1. Vista de módulos para Commander

3.3.3.3.2. Catálogo de elementos

Elemento	Responsabilidades
_models	En esta carpeta están todos los modelos que son usados por la clase de servicios. Tiene la definición de las clases en Typescript junto a sus atributos. Por ejemplo, User.
_pages	Este paquete contiene las principales páginas dentro de la aplicación. Por ejemplo el *dashboard*, explorar, *login*.
_components	Aquí están los componentes que se reusan dentro de las páginas, por ejemplo, la barra de navegación y la barra lateral.
_services	En este paquete están los servicios que son utilizados por las páginas, por ejemplo el servicio de *login* (que se encarga de la comunicación y recepción del *login* con el *backend*), el servicio general (que se encarga de hacer el resto de requests al *backend*) y el servicio de caché (que se encarga de guardar la información del caché, fijarse si está y/o eliminarla.
charts	En esta carpeta están todas las gráficas utilizadas en el *dashboard* y en las páginas, usando chartjs. Ejemplos son las gráficas de comparación de tiendas, las que están dentro de explorar Mercado Libre, y las del *dashboard*.

Tabla 3.3.3.3.2.1. Catàlogo de elementos de Commander

3.3.3.3.3. Decisiones de Diseño

Para hacer las carpetas y la organización de esta parte de la solución, se separó la lógica en las páginas, las clases de servicio y los modelos. Adicionalmente, se realizó una separación en cuanto a los componentes que usan las páginas con las páginas mismas, y se separaron las gráficas utilizadas ya que eran un módulo demasiado grande para estar dentro de components.

Para la creación de cada carpeta dentro de los paquetes, se utilizó el comando de "*ng generate component*" el cual, siguiendo con la filosofía de Angular, creaba cuatro archivos. Uno para el html, otro para el css, otro para las definiciones y otro para el código de typescript. Esto se siguió para todas las partes de la solución con excepción de services, donde se usó el comando de servicios que no incluye html ni css.

Existe un archivo no mostrado en la representación primaria llamado "*app-routing.module.ts*" donde están definidas las rutas y los componentes a los que se routean. Cada uno de estos componentes está compuesto por otros. En este contexto, por ejemplo, la página de *dashboard* es un componente, que tiene en su html la definición del componente de la barra de navegación y sus gráficos. Todos estos componentes se cargan al cargar el *dashboard*.

Al inicializarse cada componente, además, se suscribe como *observer* a los *subject* de la capa de servicios, y ejecuta los métodos de su interés. Estos métodos hacen llamadas HTTP al *backend*, y cuando el *backend* les responde, notifican al *observer* del componente. El componente, al ser notificado, carga la información y se la muestra al usuario.

4. Proceso de desarrollo

4.1. Proceso y ciclo de vida

4.1.1. Enfoque general de calidad

El enfoque general de calidad que el equipo intentó perfilar para el desarrollo fue uno principalmente basado en la revisión y mejora constante de los procesos. Esto se decidió en base a que el proyecto posee un aspecto ágil, junto con la poca experiencia que el equipo contaba para gestionar proyectos. Se esperaba mejorar los procesos de desarrollo y gestión a medida que el proyecto fuese llevado a cabo.

4.1.2. Ciclo de vida

Luego de analizar el contexto de desarrollo de la solución, el equipo optó por un ciclo de vida evolutivo[8] inspirado en metodologías ágiles. Esto fue decidido así tomando en cuenta que el equipo es más pequeño de lo usual para un desarrollo (dos personas), y los requerimientos del producto no se encuentran claramente especificados por parte del cliente.

Este ciclo de vida evolutivo permite ajustar el desarrollo y forma de trabajo según el *feedback* que se obtiene al final de cada ciclo. También permite al equipo ajustar y definir funcionalidades conforme el cliente y los usuarios vayan esclareciendo sus necesidades con respecto a las funcionalidades del producto hipotético final.

4.1.3. Metodologías de referencia

Para la elección de metodologías de referencia el equipo tuvo en cuenta mayormente aquellas con las cuales ya se hubiese trabajado. Adicionalmente, se tuvo en cuenta el tamaño del equipo y la situación inicial del proyecto para definir una metodología de trabajo.

4.1.3.1. SCRUM

“Scrum es un marco de proceso que se ha utilizado para gestionar el trabajo en productos complejos desde principios de la década de 1990. Scrum no es un proceso,

técnica o método definitivo. Más bien, es un marco dentro del cual puede emplear varios procesos y técnicas. Scrum deja en claro la eficacia relativa de la gestión de sus productos y las técnicas de trabajo para que pueda mejorar continuamente el producto, el equipo y el entorno de trabajo." - Schwaber and Sutherland[9]

Scrum fue seleccionado como referencia debido al tamaño del equipo y al aspecto ágil del desarrollo.

4.1.3.2. Kanban

"Kanban es un marco popular utilizado para implementar el desarrollo de software ágil. Requiere comunicación en tiempo real de la capacidad y total transparencia del trabajo. Los elementos de trabajo se representan visualmente en un tablero kanban, lo que permite a los miembros del equipo ver el estado de cada trabajo en cualquier momento." [10]

Kanban es utilizado en el proyecto para representar visualmente el estado de los *sprints*, ayudando a informar a todos los integrantes del equipo del estado del proyecto.

4.1.4. Descripción general del proceso

Dado que el equipo consta de solamente dos personas que trabajan en cercanía con Astroselling, se posee un contacto contìnuo con el cliente. Los requerimientos no se encuentran completamente definidos, tendiendo a fluctuar a lo largo de la ejecución del proyecto. Por lo tanto, se decantó por utilizar una metodologìa de trabajo ágil basada en SCRUM, incorporando elementos de Kanban.

El objetivo para la metodología de trabajo es implementar el modelo PDCA (Plan Do Check Act) [11] a lo largo de todo el proceso de desarrollo. Esto se hace con la intención de poder avanzar en ciclos iterativos de trabajo con mejora continua. Se espera que al ir obteniendo métricas a partir de ciclos completados, esta información será tomada en cuenta en el planeo de ciclos posteriores. Las actividades en las que se puede ver reflejado esto son el sprint planning (Plan), la ejecución del *sprint* (Do), *sprint reviews* con *sprint retrospectives* (Check) y finalmente las estimaciones y *planning* subsecuentes (Act).

Los marcos temporales de trabajo se dividieron en *sprints*, cada uno de estos abarcó dos semanas. En una primera instancia se utilizó el concepto de historias de usuario, con tal de poder establecer una validación que mostrara cuál era el valor que se estaba agregando al usuario con esa funcionalidad. En etapas posteriores del desarrollo el formato de historia de usuario fue cayendo en desuso, dado que la naturaleza de las tareas fue lentamente tornando a una naturaleza más de soporte y correcciones al producto.

Estas tareas se estimaron en *story points* para establecer prioridades dentro de cada *sprint*. Las tareas en su totalidad comenzaron estando en el *product backlog*, y en cada *sprint* se eligieron tareas para realizar de acuerdo a las necesidades y los requerimientos del cliente. Se tomaron dos semanas para la duración del *sprint* ya que se evaluó que es un tiempo apropiado para realizar chequeos en el avance del proyecto, y menos tiempo podría producir *sprints* sin avances relevantes.

Se realizó una ceremonia de seguimiento todos los lunes para evaluar el estado actual del *sprint*. Esta ceremonia fue seguida de la *sprint retrospective* los días que coinciden con la finalización del *sprint*. En una primera instancia, se consideró que la ceremonia

retrospectiva será importante, dada la particularidad de ser dos integrantes, ya que permite al equipo evaluar su desempeño en los *sprints* y ajustar detalles de la ejecución según sea necesario.

Tomando en cuenta que habrá una parte significativa del trabajo a realizarse los fines de semana, se considera adecuado dar como día de inicio/fin de los *sprints* los lunes.

Para realizar la estimación de tareas del *backlog* se utilizó *planning poker.* [12] Las estimaciones en conjunto con las horas de trabajo invertidas al final de cada *sprint* conforma parte de las métricas que son utilizadas para ajustar el *backlog* de tareas.

Estas mediciones sirvieron para ofrecer una visualización del trabajo realizado y las horas invertidas del equipo, al graficarlas en cada *sprint review.* Ambos integrantes del equipo consideran que el *planning poker* genera discusiones productivas al momento de evaluar y estimar las tareas a realizar en el *sprint*, a pesar de que serán más leves dado que el equipo se compone de únicamente dos integrantes.

4.1.5. Medición y seguimiento del proceso

Debido a la metodología de desarrollo ágil, los procesos utilizados cuentan con una naturaleza más reactiva que proactiva. Esto significa que ante adversidades y situaciones fuera de lo planeado, es preferible adaptar los planes a perdurarlos. Esto genera la cuestión de obtención y análisis de métricas, dado que lo que no se mide no se conoce, lo que no se conoce no se puede cambiar y lo que no se puede cambiar no se puede gestionar.

Con este objetivo en mente, durante las iteraciones del desarrollo se mide, con la ayuda de herramientas de gestión, dos métricas para incorporar en el *loop* de desarrollo.

La primera métrica es puntos de trabajo completados por *sprint*, también conocida como la velocidad del equipo en contextos de desarrollo ágil. [13] Esta métrica permite saber la capacidad de trabajo que el equipo puede llevar a cabo por semana, y ayuda así a estimar fechas para funcionalidades a implementar, brindando un seguimiento a Astroselling de esta manera.

La segunda métrica es puntos de trabajo por hora por *sprint*. El equipo se refiere a esta métrica como la eficiencia del equipo, y permite asignar cargas de trabajo por *sprint* apropiadas para el equipo.

4.1.6. Evolución del proceso durante el proyecto

4.1.6.1. Comienzo e investigación inicial

A comienzos del proyecto el equipo realizó un *proof of concept* junto con *spikes* investigativos con una duración de dos meses hasta llegar a un prototipo funcional del requerimiento de recolección de datos. Esto permitió validar la posibilidad de concretar una aplicación que analice todo Mercado Libre con cierta frecuencia.

Dentro de este proceso fueron considerados detalles como el tamaño de los archivos resultantes del scrapping de mercado libre, así como el tiempo que duraría un proceso de esta magnitud.

Los detalles de esta investigación pueden ser encontrados en el Anexo 1 - Investigación inicial.

4.1.6.2. Desarrollo inicial

A principios del segundo cuatrimestre del proyecto, el equipo incorporó un ciclo de desarrollo evolutivo en el cual se utilizan métricas para ajustar el trabajo a través de los *sprints* con una metodología PDCA.

Debido a que el cliente definió un conjunto general de objetivos para Astrolytics pero no ha brindado requerimientos detallados para la aplicación, el equipo concluyó que sería ideal los procesos en base al modelo de prototipo evolutivo. Esto también ayudará con el hecho de que el equipo no poseía experiencia previa significativa con respecto a gestión de equipos y desarrollos. De esta manera, la evolución no sería solamente del prototipo en sí, sino también de los procesos relacionados con su desarrollo. [8]

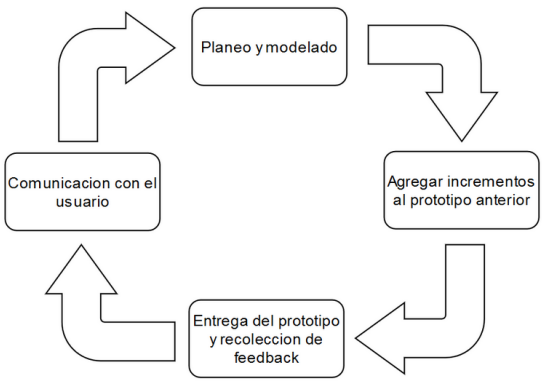


Imagen 4.1.6.2.1. Esquematización del modelo de procesos general

4.2. Gestión de proyecto

A continuación se describen varios aspectos de gestión llevados a cabo en el proyecto.

4.2.1. Gestión del *Backlog*

El relevamiento de requerimientos se realizó intentando maximizar el involucramiento del usuario, con el uso de entrevistas junto al cliente. A partir de la información recabada en estas entrevistas, se convierten los requerimientos funcionales a *user stories* que pueden ser encaradas por el equipo de desarrollo.

El seguimiento de las tareas a realizar en cada *sprint* fue realizado en la herramienta de gestión de equipos Trello. En esta se posee un *board* al estilo Kanban, donde se realiza un seguimiento de las *user stories* y tareas del *backlog*, estimarlas, seguir su ejecución y limpiarlas.

4.2.2. Planificación temporal y de recursos

El equipo de desarrollo formalizó la estructura temporal del proyecto a inicios del 2020, comenzando a operar en *sprints* de dos semanas, y realizando evaluaciones en la finalización de cada uno. En estas instancias de evaluación, el equipo distribuye sus recursos temporales (horas de trabajo) en las distintas tareas a asignar desde el *backlog*, mientras que también reserva una cantidad de horas de trabajo para imprevistos.

El objetivo de horas semanales comenzó en 20 horas individuales por *sprint*, y fueron gradualmente elevadas hasta mantenerlas en 40 horas luego de algunas iteraciones.

4.2.2.1. Gestión de horas de trabajo

Para llevar una mejor trazabilidad y seguimiento del esfuerzo realizado, se realizó un seguimiento de las horas de trabajo día a día para cada uno de los integrantes del equipo, manejando los conceptos de las horas. Estas son utilizadas más adelante para realizar métricas de trabajo.

Se eligiò, bajo recomendaciòn del tutor, la página Toggl.com para llevar a cabo el seguimiento de las horas. Se creó un proyecto nuevo llamado Astrolytics y es allí donde se cargarán las horas a medida que se hagan, para los dos integrantes del equipo.

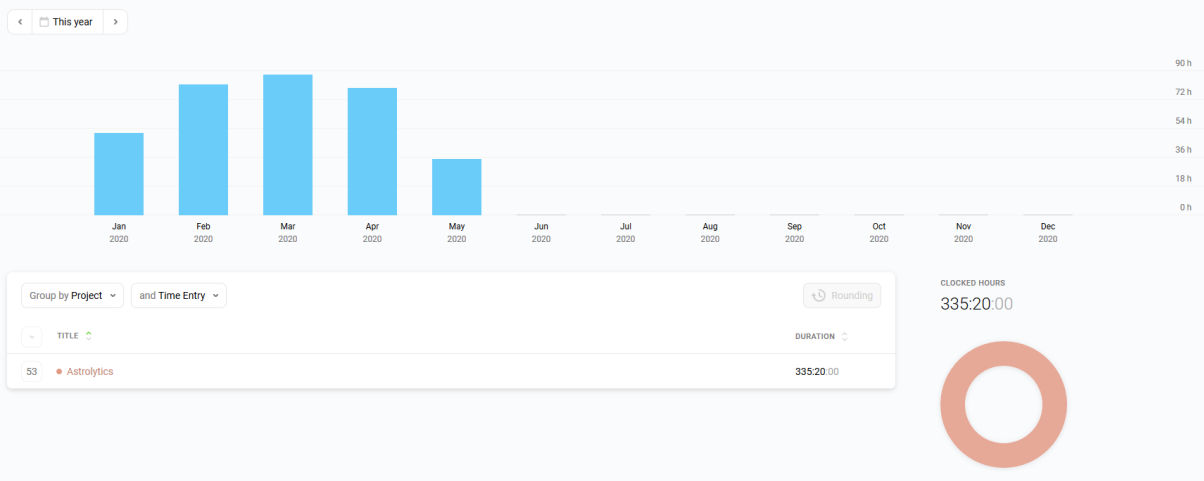


Imagen 4.2.2.1.1. Registro de horas de trabajo en Toggl

4.2.2.2. Línea de tiempo del proyecto

A continuación se presentan los hitos que han conformado el desarrollo de Astrolytics, desde su comienzo el 10 de septiembre del 2019, hasta su finalización el 10 de septiembre del 2020.

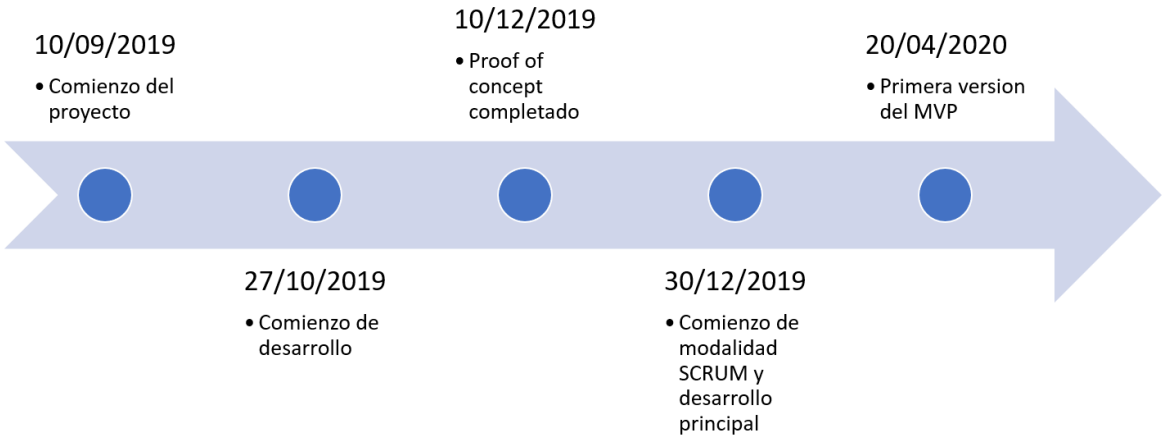


Imagen 4.2.2.2.1. Línea de tiempo del proyecto, año 2019

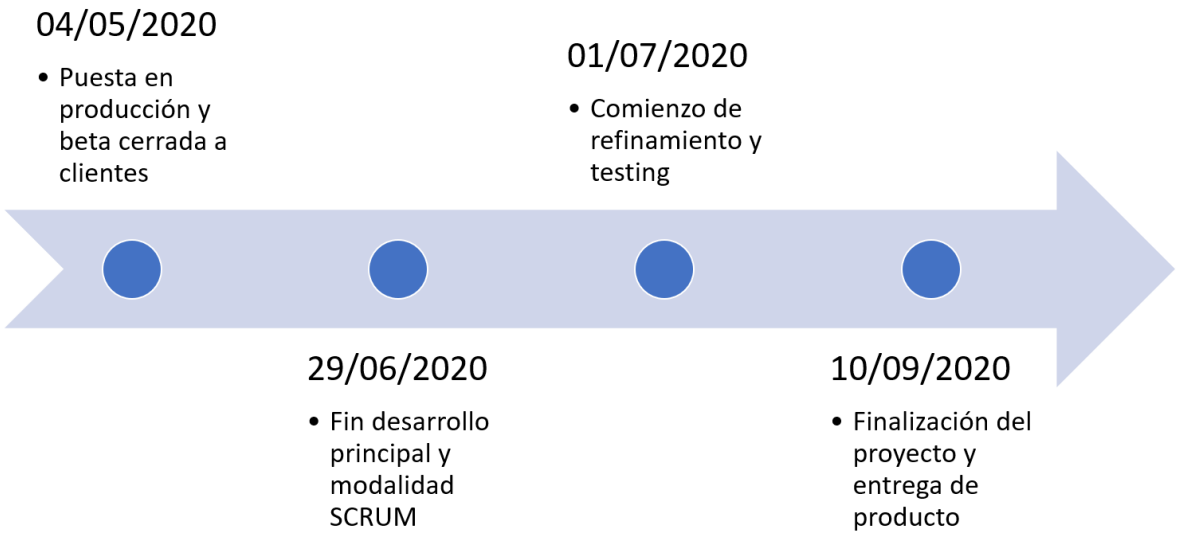


Imagen 4.2.2.2.2. Línea de tiempo del proyecto, año 2020

4.2.3. Seguimiento y evaluación de iteraciones

Como fue mencionado previamente, el equipo adoptó una metodología que involucra iteraciones de dos semanas. El seguimiento del *sprint* es realizado en Trello, en el *board* del equipo. Al final de cada iteración, el equipo ejecuta las ceremonias de *sprint review*[9] y *sprint retrospective.* [9]

4.2.3.1. *Sprint Retrospective*

En el *sprint retrospective* el equipo de desarrollo observa los resultados del *sprint* pasado, tomando en cuenta los detalles de las horas anotadas y cómo se relacionan estas con las tareas completadas en el *sprint*, tratando de incorporar a la retrospección detalles personales como sucesos, responsabilidades, disponibilidad horaria propia, etc. A partir de este análisis, el equipo realiza un punteo con el objetivo de identificar los detalles del *sprint* que se consideran salieron bien, así como potenciales mejoras a tener en cuenta para el próximo *sprint*.

4.2.3.2. *Sprint Review*

Durante los *sprint reviews* se realiza una puesta en escena del incremento realizado en el *sprint* anterior. Adicionalmente, se incorpora al análisis el *feedback* de los clientes realizados durante el *sprint* y el estado actual del *backlog*. El objetivo de esta reunión es ajustar el *backlog* del producto acorde a la situación actual, priorizando tareas y creando otras nuevas, con la intención de otorgar la mayor cantidad de valor posible al cliente.

4.2.4. Comunicación y coordinación del equipo

La coordinación del equipo con respecto al trabajo asignado se realizó mayormente los lunes, en el *sprint planning*. En esta reunión se establecieron los puntos a afrontar con respecto al desarrollo y qué integrante se haría cargo de cada uno.

Se realizaron reuniones de seguimiento del proyecto con el tutor donde se discutieron aspectos relacionados con gestión y consideraciones a tener en cuenta con el proyecto como trabajo final de carrera.

Con respecto a la comunicación del equipo, se mantuvo una vía de comunicación abierta constante con el uso de WhatsApp, pudiendo contactarse los integrantes del equipo en cualquier momento que fuese necesario. La comunicación con el cliente fue mayoritariamente a través de Slack, con uso de mails también en caso de requerir un intercambio de índole más formal y concreta, así como también se realizaron reuniones informales de manera semanal para discutir el progreso de la aplicación. Es importante notar que estas reuniones eran facilitadas por el hecho de que ambos integrantes trabajaban en conjunto con el cliente.

4.2.5. Gestión del riesgo

Debido a la naturaleza ágil e incierta del desarrollo en el proyecto de Astrolytics, los integrantes concluyeron que los riesgos del proyecto se lidiaran de dos maneras distintas: proactivamente y reactivamente. Con respecto al manejo reactivo de los riesgos, este se debe a que se entiende que en un proyecto no se puede prever todo tipo de riesgo que pueda llegar a incurrir, por lo tanto semanalmente se dedica un tiempo de trabajo al propósito de solucionar los problemas que puedan surgir en el camino.

El manejo proactivo de los riesgos fue realizado en etapas, identificación, proyección y refinamiento de riesgos. Al momento de realizar la evaluación de riesgos del proyecto se tuvo en cuenta los siete principios de gestión de riesgos planteados por el Software Engineering Institute. [8]

4.2.5.1. Identificación

Para realizar la identificación de riesgos se tuvo una reunión junto al cliente con la intención de realizar una lluvia de ideas con respecto a qué riesgos podrían presentarse a lo largo del desarrollo del producto y la ejecución del proyecto. El resultado de esta reunión se dio como una lista de riesgos, sin considerar que tan posibles o no eran aún. Esta lista pasó a ser usada en la siguiente etapa.

- Mercado Libre podría estar en contra del desarrollo de la aplicación, y por ende bloquearla de su dominio o no permitirnos obtener información de alguna manera.

- Alguno de los integrantes del equipo podrían encontrarse en condiciones inoperantes, ej hospitalizados.
- Podría filtrarse información sensible de los usuarios de la aplicación.
- Los usuarios podrían encontrarse desinteresados y no dispuestos a ayudar y proveer feedback.
- El equipo de desarrollo podría encontrar dificultades al momento de trabajar con nuevas tecnologías.

Es importante notar que esta lista no es exhaustiva en absoluto, dado que es imposible prever todos los riesgos existentes.

4.2.5.2. Proyección

En la etapa de proyección, el equipo se encargó de clasificar los riesgos identificados y listados en la etapa de identificación. Estos fueron clasificados dentro de dos dimensiones. probabilidad e impacto. El objetivo de esto consta de poder priorizar los riesgos y determinar qué recursos serán utilizados para afrontar cada uno de ellos, dado que los recursos del proyecto son limitados.

La escala de probabilidad es porcentual y habla de la probabilidad de que el riesgo hipotetizado ocurra. El impacto es clasificado dentro de cuatro categorías: catastrófico, crítico, marginal y despreciable. Catastrófico describiendo la falla total del proyecto, y despreciable siendo un riesgo completamente ignorable.

Riesgo	Probabilidad	Impacto
Mercado Libre podría estar en contra del desarrollo de la aplicación, y por ende bloquear la aplicación de su dominio o no permitirnos obtener información de alguna manera.	60%	Catastrófico
Alguno de los integrantes del equipo podría encontrarse en condiciones inoperantes, ej hospitalizado.	10%	Catastrófico
Podría filtrarse información sensible de los usuarios de la aplicación.	30%	Crítico

Los usuarios podrían encontrarse desinteresados y no dispuestos a ayudar y proveer *feedback*.	30%	Crítico
El equipo de desarrollo podría encontrar dificultades al momento de trabajar con nuevas tecnologías.	40%	Marginal

Tabla 4.2.5.2.1. Proyección de riesgos

4.2.5.3. Mitigación, monitoreo y gestión

A partir de la tabla de clasificación de riesgos de la sección anterior, fueron identificados los riesgos más imperativos a confrontar.

El primer riesgo que el equipo gestionó fue la aprobación de Mercado Libre con respecto al desarrollo de la aplicación. Para mitigar este riesgo el equipo se puso en contacto con un representante de Mercado Libre, Javier Villalba, explicándoles el alcance de la solución y preguntando qué recursos de la página podrían usarse en caso de que no quisieran ser consultados de manera atípica. Afortunadamente, Mercado Libre respondió afirmativamente al desarrollo, y hasta proporcionó varios *endpoints* que ellos consideraron podían resultarnos útiles. Los detalles de esta interacción pueden encontrarse en el Anexo 1 - Investigación inicial, en el dia 28/10 y 01/11 de la investigación inicial del proyecto.

Con respecto a las dificultades técnicas y de desarrollo, se consultó con el tutor del proyecto, el cual comentó que el equipo tendría suficiente tiempo para o aprender la tecnología nueva, o migrar un módulo a un lenguaje mejor manejado por el equipo en caso de ser necesario, gracias a que el equipo se encontraba en una etapa temprana del desarrollo. Con esta información, el equipo se ocupó de realizar un seguimiento constante a la productividad, con tal de detectar la posibilidad de que se estuviera generando un cuello de botella a causa de dificultades.

El desinterés de los usuarios no ocurrió de manera esperada, sino a causa de la pandemia global que afectó el año de 2020. En la siguiente sección del documento se puede leer cómo se afrontó este problema.

4.2.6. Análisis de problemas de gestión

4.2.6.1. Cambio de lenguaje en un módulo

A medida que avanzó el desarrollo, el equipo observó atentamente el progreso en cada uno de sus módulos. A mediados de enero se noto que el avance en el modulo de analitica se estaba dificultando por el desarrollo en NodeJS, debido a que las tareas se estancaron mayormente por dificultades técnicas relacionadas con el lenguaje y no tanto por la dificultad de la lógica a implementar. Bajo recomendación del tutor, el equipo decidió migrar el desarrollo actual del módulo a otro lenguaje. Python fue la elección final, debido a que ofrece el *framework* Flask, que permitió migrar todas las funcionalidad actualmente implementadas desde el módulo existente. También aportó mucho a la elección la comodidad y familiaridad de desarrollo que ambos integrantes poseen con el lenguaje.

Este fue el único evento del desarrollo de la aplicación donde se pudo observar un cuello de botella causado por dificultades técnicas.

4.2.6.2. Problemas de comunicación en el comienzo

El equipo de desarrollo obtuvo luz verde para el proyecto de Astrolytics en septiembre de 2019. Una vez fue concretado el proyecto a desarrollar, el equipo comenzó a realizar investigaciones y pruebas de concepto para confirmar que el desarrollo de la aplicación supuesta era posible. Este periodo no fue gestionado con la misma rigurosidad y formalidad que el resto del proyecto pero fue documentado en una bitácora la cual será anexada al final del documento actual.

Esta falta de gestión se dio a ver en el primer informe de avance del equipo, lo cual incitó al tutor y sus integrantes a reveer la organización y gestión del trabajo que se estuviese realizando.

Así, en diciembre del 2019, el equipo comenzó a utilizar *sprints* para organizar su trabajo, junto con un *backlog* para organizar las tareas de desarrollo y herramientas de gestión de tiempo con un tiempo de trabajo objetivo semanal para cada integrante.

4.2.6.3. Eliminación de tokens

Inicialmente, la aplicación almacenaba en su base de datos un hilo de caracteres conocido como el Mercado Libre (MeLi) *token*. Este *token* permite acceso a datos privados del usuario, los cuales eran utilizados para ofrecer estadísticas y métricas de venta personalizadas. Sin embargo, este *token* también permite administrar las publicaciones y pagos del usuario, incluyendo eliminación de publicaciones, por lo que es un dato altamente sensible. Luego de un análisis del equipo, se decidió eliminar estas funcionalidades y no almacenar el *token* en la base de datos, debido a que se consideró que el valor que aportan no compensaba el riesgo de seguridad que implicaba el almacenamiento y administración de la clave.

4.2.6.4. Pandemia global

El 11 marzo de 2020 fue declarada por la Organización Internacional de la Salud (WHO) una pandemia global de COVID-19. [14] Entre las consecuencias a nivel cultural y estatal se pudieron encontrar regulaciones y recomendaciones de distanciamiento social, junto con cuarentenas obligatorias u optativas.

Este evento tuvo su efecto en el desarrollo de Astrolytics. Por un lado, toda reunión presencial debió ser cancelada inmediatamente y la compañía presencial diaria de Astroselling fue perdida debido a que se efectuaron medidas para implementar el trabajo de modalidad "*Home Office*". Por otro lado, los usuarios no solo se encontraban indispuestos a realizar reuniones con el equipo de desarrollo por las medidas preventivas, sino que también varios se encontraban en cese o reducción de operaciones y ventas, por lo que no se encontraban dispuestos a brindarle tiempo al desarrollo de Astrolytics.

Una vez Astroselling se encontró en condiciones más estables en abril, se dedicó tiempo a una reunión para elaborar medidas en respuesta a la pandemia. Entre estas se pudo encontrar la realización de reuniones a distancia con tal de mantener el contacto con el cliente a lo largo del desarrollo. También, para volver a involucrar a los usuarios en el desarrollo de la aplicación, se incluyó al *backlog* una funcionalidad que les permitiese proveer *feedback* directamente desde la aplicación. Esto le permitiría a los usuarios poder brindar sus opiniones con respecto a la aplicación,

aportar sugerencias o críticas sin necesidad de contactar directamente con nosotros. Detalle de estas sugerencias puede encontrarse en el Anexo 5.

Con respecto a la organizacion y gestion interna al equipo de desarrollo no han habido cambios sustanciales, debido a que las reuniones para intercambiar actualizaciones y planear el desarrollo ya se realizaban con modalidad no presencial.

4.3. Ingeniería de requerimientos

4.3.1. Técnicas y resultados de investigación y especificación

El desarrollo se realizó a partir de dos fuentes principales de requerimientos. La primera fuente es el cliente Astroselling, y la segunda son los usuarios del prototipo de Astrolytics, que, en esta instancia, son un subconjunto de los usuarios de Astroselling.

Los requerimientos se obtendrán a lo largo de todo el proyecto, ya que dado el ciclo de desarrollo y la metodologìa de trabajo basada en SCRUM, habrá contínuamente nuevas versiones del sistema y con ella, nuevas ideas y sugerencias nacidas del *feedback* de los principales *stakeholders*.

4.3.1.1. Entrevistas

Esta metodología es la que se utilizó con los usuarios finales del prototipo. A lo largo del desarrollo se pidieron reuniones con diferentes agentes y a través del diálogo y la atención a sus necesidades, se anotaron requerimientos funcionales y no funcionales que luego pasaron a ser historias de usuario las cuales fueron agregadas al *backlog.*

En un principio se han organizado entrevistas con tres clientes que Astroselling considera serían una muestra apropiada para el modelo de usuario final de la aplicación y demostraron interés en una funcionalidad de metricas y analiticas para su plataforma.

La metodología para las entrevistas consistió en tener listo un *wireframe* de la aplicación desarrollado en Balsamiq, a través del cual presentandolo al cliente se fueron describiendo las ideas y objetivos de la aplicación, dialogando con ellos en conjunto.

4.3.1.1.1. CalzadosUY

La primera entrevista fue realizada el 27 de diciembre de 2019 con CalzadosUY, un cliente de tamaño medio-pequeño que opera sobre Mercado Libre ofreciendo una modesta gama de calzados. Fueron validadas varias de las funcionalidades que surgieron inicialmente, también se confirmaron que otras aportan menos valor del que

fue supuesto inicialmente y surgieron varias funcionalidades que en el análisis inicial no se habían considerado. También se presentó el caso de uso de una empresa que podría no estar actualmente vendiendo en Mercado Libre y desea hacer un análisis previo con Astrolytics herramienta para evaluar si ingresar al mercado digital.

Cada uno de los integrantes del equipo tomó sus notas de la entrevista y luego fueron discutidas en una reunión posterior entre ambos, identificando y refinando los requerimientos para presentarlos finalmente a Astroselling y agregarlos al *backlog* como *user stories*. Es posible encontrar las notas de la entrevista en el Anexo 3.

4.3.1.1.2. CLEVER

La segunda entrevista presencial tomó lugar el 11 de marzo del 2020 con Clever, una empresa de *ecommerce* de tamaño medio especializada en variedad de productos, la cual es actualmente cliente de Astroselling. Es importante notar que esta entrevista ocurrió dos días antes de la declaración de emergencia sanitaria nacional. [14]

La entrevista consistió en una explicación de la herramienta Astrolytics al entrevistado, primero recopilando opiniones sobre si la herramienta resultaba atractiva a nivel conceptual. Luego se presentó la aplicación con el progreso realizado hasta el momento, ofreciendo un tour de las funcionalidades programadas. Se brindó al usuario el detalle de cuáles eran los datos presentados, y que capacidades posee la aplicación momentáneamente. Finalmente se preguntó por la opinión general del entrevistado, junto con detalles que ellos consideran importantes dar a conocer y que les interesaría ver en un futuro para la herramienta.

El modus operandi del equipo fue el mismo que la entrevista anterior a CalzadosUY, refinando detalles de la entrevista para ingresarlos al backlog posteriormente. Se obtuvo información valiosa, la cual sirvió para validar funcionalidades y conceptualizar otras a implementar.

También se presentó la posibilidad de invitar al entrevistado a una beta cerrada de la aplicación en un futuro, a la cual accedieron amablemente.

4.3.1.2. Brainstorming

La tormenta de ideas para obtener requerimientos es un proceso que ha sido utilizado por el equipo en proyectos anteriores, la principal contraparte en el *brainstorming* será el cliente, Astroselling. A través del intercambio de ideas la intención es obtener nuevos requerimientos que después se pudiesen refinar y validar con la ayuda del cliente.

4.3.2. Relevamiento de requerimientos no funcionales y restricciones

Con respecto a la inferencia y relevamiento de requerimientos no funcionales, el equipo tuvo en cuenta los atributos de calidad enumerados en Software Architecture in Practice[6] para clasificarlos. Debido a que el cliente no fue explícito en aspectos referentes a detalles del sistema como *performance* o mantenibilidad, el equipo se encargó de ajustar estándares de funcionamiento en forma de requerimientos de atributos de calidad y restricciones del sistema.

4.3.3. Justificación de la estrategia de relevamiento

La estrategia principal para el relevamiento estuvo conformada mayormente de entrevistas y consultas a los usuarios de Astroselling debido a la naturaleza poco definida del producto final. Se consideró importante estar seguro de que el equipo estuviese desarrollando un producto que le brinde valor a los usuarios, y no uno sin sentido. Por esto fue realizado un énfasis importante en la comunicación y recolección de *feedback* con el usuario para poder validar los requerimientos a desarrollar.

De esta manera, al estar mostrando al usuario y al cliente el progreso de la aplicación, se ayuda a que ellos pudieran esclarecer sus dudas y concretar sus necesidades. Este *feedback* obtenido de las demostraciones es luego incorporado al ciclo de desarrollo en las reuniones de *sprint review* y utilizado para ajustar el *backlog* y carga de trabajo del próximo *sprint*.

4.3.4. Conceptualización del problema y su contexto

El problema a resolver que Astroselling planteó al equipo fue “¿Cómo podemos solucionar la necesidad de métricas de negocio de nuestros usuarios?”. Esto generó

dos contextos a tener en cuenta de manera inmediata, el primero consiste en que no se posee un problema o situación específica o detallada para solucionar, sino que era un planteo más abierto y menos definido. El segundo contexto que levantó esta situación fue que Astroselling no posee un solo tipo de usuario, lo cual significaba que se tendría que contemplar muchos casos de uso y situaciones distintas, para poder satisfacer las necesidades de información y métricas de los distintos usuarios de Astroselling.

4.3.5. Criterios de la priorización de los requerimientos

Los requerimientos fueron priorizados de manera dinámica conforme el desarrollo fue avanzando. En una primera instancia se priorizaron los requerimientos necesarios para obtener un MVP mostrable a usuarios. Luego de este hito, la priorización de los requerimientos cambió para valorar más el agregar funcionalidades que los usuarios pidan de manera más explícita.

4.3.6. Prototipos realizados

Con prototipo, se refiere a una versión que ofrece una mínima funcionalidad en un área de interés que permite simular en cierta parte la experiencia final que va a tener el usuario con la herramienta a desarrollar.

Los prototipos son facilitados por la metodología ágil de desarrollo utilizada, a través de ellos y la experimentación de parte de las partes interesadas se obtuvo *feedback* sobre mejoras o nuevas funcionalidades que se convertirán en requerimientos.

Para la muestra de incrementos y obtención de *feedback* por parte del cliente se utilizaron varios prototipos que fueron volviéndose más funcionales conforme avanzaba el desarrollo.

4.3.6.1. MVP

El primer objetivo fue realizar un MVP de la aplicación, el cual pudiera ya desde un comienzo proporcionarle un valor agregado al usuario, por leve que pudiera ser. Una versión preliminar de este MVP fue completada el 20 de abril del 2020, y se mostró en funcionamiento durante una reunión con Astroselling y con un cliente selecto más.

El resultado de esta reunión fue sumamente positivo, con los representantes de Astroselling y el cliente sugiriendo correcciones y mejoras posteriores.

El MVP en un ambiente accesible desde afuera será el resultado final entregable para Astroselling al final del proyecto.

4.3.6.2. Beta

El 5 de mayo de 2020 el equipo de desarrollo lanzó la beta cerrada de Astrolytics. Astroselling eligió un grupo de usuarios con los que poseían contacto frecuente y que habían identificado como interesados en la aplicación para que participaran en la beta cerrada y pudieran proporcionar *feedback* luego de 2 semanas de uso. Es importante notar que para el lanzamiento de la beta fue desarrollada una funcionalidad de *feedback* en la aplicación misma, con el objetivo de facilitar e incentivar a los usuarios a opinar sobre el producto. El resultado de esta funcionalidad puede encontrarse en el Anexo 5, donde se muestran varias capturas de las sugerencias realizadas por los usuarios de la beta.

4.3.7. Estructura utilizada para la especificación

Para la especificación de requerimientos funcionales nuevos se utilizó la estructura de *user stories* desarrollada en “User Stories Applied” escrito por Mike Cohn, teniendo en cuenta las características que propone conforman una *user story* bien formada, comúnmente identificada con la sigla INVEST. (Independiente, Negociable, Valiosa, Estimable, Pequeña y Testeable) [15]

Para la formulación del contenido de la historia se utilizó también una narrativa que describe quien es el usuario, cuál es la funcionalidad a desarrollar y el beneficio que esta funcionalidad le proporciona al cliente. Luego se pueden añadir notas al pie de la tarjeta con información que pueda ser relevante para el desarrollo. [16]

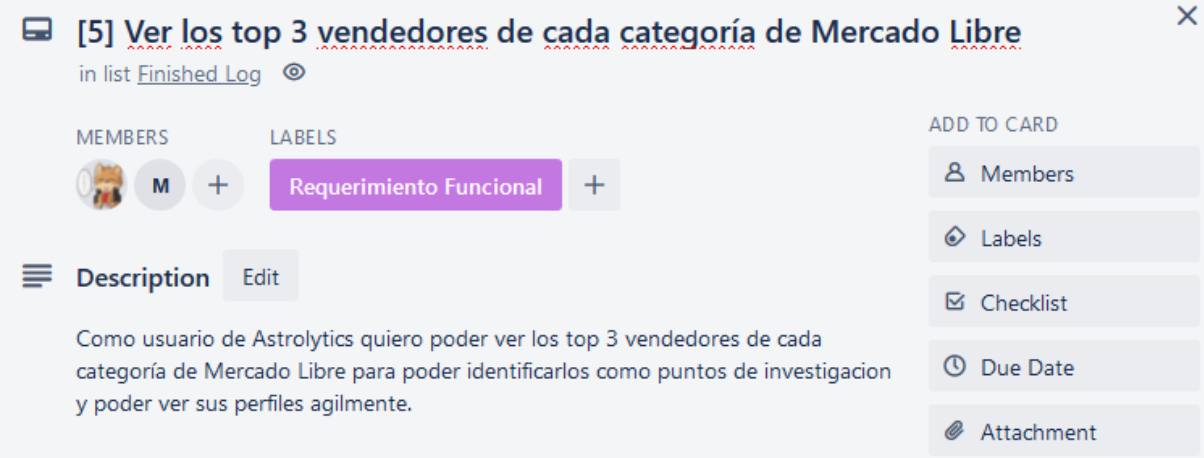


Imagen 4.3.7.1. *User story* utilizada en Trello

Esta estructura permite al equipo trabajar con las tarjetas en el *board*, discutirlas y estimarlas de manera simple y tener claro cuál es el incremento que aporta cada una de ellas. En el título, entre paréntesis rectos se tiene la estimación del tamaño en *story points*, se utilizan *labels* también para poder categorizarlas, y se tiene una descripción detallada si se abre la tarjeta.

En el caso de funcionalidades pedidas por parte del cliente, *bugfixes* o comodidades del programador, el formato de *user story* no fue utilizado, debido a que no era necesario realizar una validación del valor de la tarea. Esto resultó en que a finales del proyecto el formato de *user story* ya no fuera utilizado, dado que no se estaban implementando tantas funcionalidades nuevas que requerían una validación adicional de su valor al usuario.

4.3.8. Validaciones realizadas a la especificación

Las validaciones de los requerimientos implementados fueron pensadas con un énfasis fuerte en que los clientes lo aprobaran.

4.3.8.1. Wireframe

En instancias muy tempranas de desarrollo, puntualmente dentro de los primeros tres meses, se realizó un *wireframe* tomando en cuenta las funcionalidades básicas que se pudieron inferir hasta el momento. Esto fue hecho con el objetivo de validar la especificación temprana y contar con la confirmación de que el equipo desarrolla un MVP que pudiera ofrecer mínimo incremento esencial para el cliente.

El *wireframe* creado fue el siguiente.

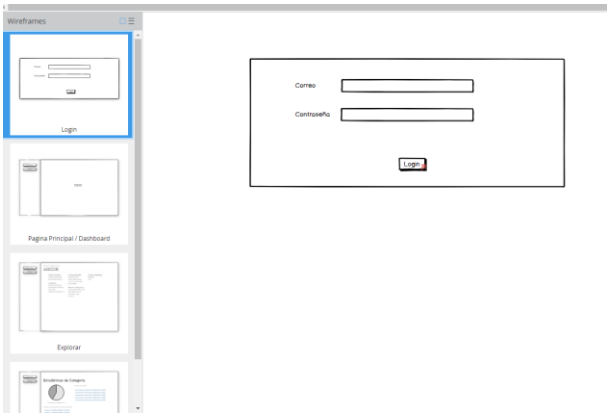


Imagen 4.3.8.1.1. *Wireframe* para el *login* de la aplicación

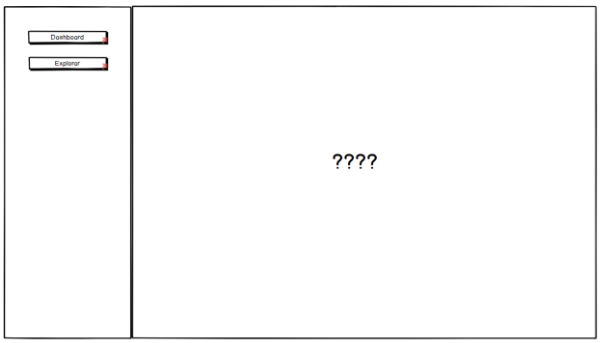


Imagen 4.3.8.1.2. *Wireframe* para la barra lateral de navegación

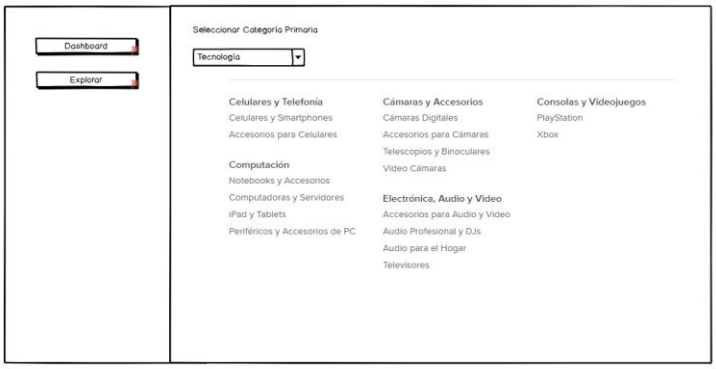


Imagen 4.3.8.1.3. *Wireframe* para la exploración de categorías

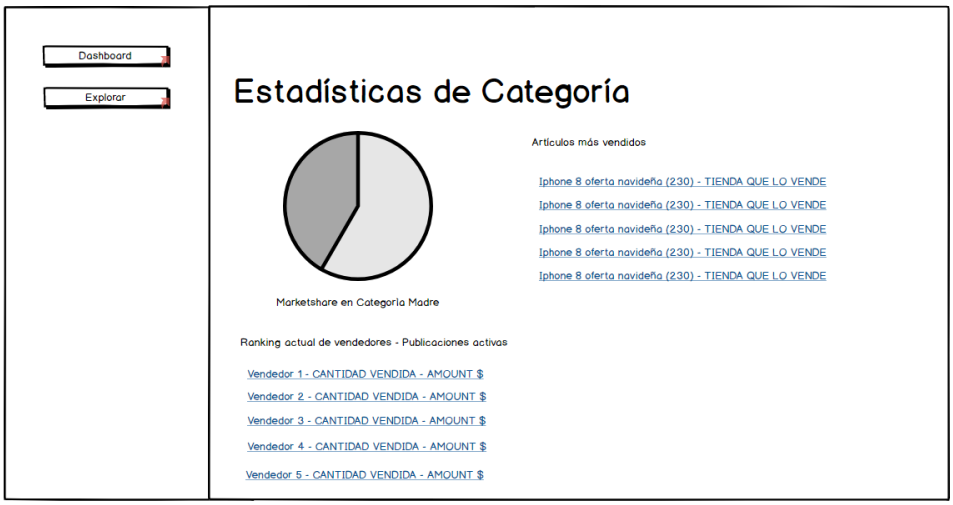


Imagen 4.3.8.1.4. *Wireframe* para el detalle de categoria

Es importante notar que las validaciones a la especificación son realizadas de manera repetitiva a lo largo del desarrollo, validando y confirmando el valor de las funcionalidades implementadas conforme sea posible.

4.4. Diseño de la arquitectura

A continuación se puede encontrar descrito los detalles arquitectónicos del desarrollo, desglosado en cada uno de los componentes de la solución completa.

4.4.1. Introducción

El diseño de la arquitectura utilizada fue uno de los desafíos más grandes a los cuales el equipo se vio enfrentado a la hora de realizar el proyecto. Al principio, se sabía que se tenía el desafío de poder scrapear toda la página de Mercado Libre Uruguay con sus más de 6 millones de publicaciones, y se quería poder realizarlo en menos de un día.

Desde la primera etapa de gestación se decidió que la responsabilidad de toda la plataforma de Astrolytics iba a estar dividida en varias partes, el motivo principal de esto era no solo el poder desacoplar el desarrollo y la implementación, si no por poder dividir la carga de la aplicación en varias partes.

El primer atributo de calidad considerado a la hora de diseñar la arquitectura, fue el de eficiencia. Como premisa, se quería tener la absoluta certeza que se iban a poder scrapear todas las publicaciones en menos de 24 horas, para poder ejecutar el proceso por lo menos una vez al día. El segundo desafío, también asociado a la eficiencia, era cómo se iba a procesar un caudal tan grande de datos en tiempo y forma, y qué tipo y modelo de base de datos se iba a utilizar para poder guardar esta información, al igual que qué tan importante era el decidir qué información se iba a guardar, balanceando entre el aporte que generaba al negocio y lo que costaba no solo el procesamiento si no el almacenamiento de esta información.

Por tal motivo, antes de empezar el proyecto en sí, se creó un prototipo que serviría como una prueba de carga, un *proof of concept* que diría si sería posible o no hacer lo que se deseaba hacer. Un análisis detallado de esta prueba para scrapear Mercado Libre se puede encontrar en el Anexo 1.

Las conclusiones fueron que utilizando *multi-threading* y la api de búsqueda de Mercado Libre, era posible obtener información de todas las publicaciones en alrededor de 2 horas, subiendo un archivo .zip que pesa hoy día alrededor de 150

megas con toda la información de categorías. Esta velocidad de scrapeo se debe en gran parte por utilizar la API oficial de Mercado Libre, sacrificando la exactitud de cierta información como la cantidad exacta vendida y el stock disponible, que es obtenida pero en rangos que permitan dar una idea de los movimientos de esas publicaciones.

La cantidad exacta vendida y de stock disponible para cada publicación está disponible a simple vista en Mercado Libre si uno visita una publicación, pero se debe de realizar un scrapeo del HTML de la página y no de la API (u obtener el *token* de sesión del usuario que ha realizado la publicación, lo cual no es viable ya que se deberían de tener los *tokens* de todos los usuarios de Mercado Libre) lo cual tiene un impacto en la eficiencia considerable, pudiendo scrapear de este modo, según el análisis realizado, tan solo aproximadamente un 10% de las publicaciones totales de Mercado Libre por día, lo cual no sirve para el propósito del proyecto.

Este prototipo creado para probar si era posible obtener la información, fue el que luego evolucionaría y se transformaría en el componente Scraper. Existen además otros dos componentes principales, llamados Analytics *y* Commander*,* que serán vistos a continuación.

4.4.2. Componentes de la arquitectura

Todo lo que se escriba a continuación es acerca de los componentes en su etapa final luego de un proceso de iteración sobre ellos y sus interacciones.

4.4.2.1. Scraper

Como se comentó en la sección anterior, este componente es el encargado de obtener la información de Mercado Libre. Utiliza la API de Mercado Libre para buscar todas las publicaciones de una categoría, y va guardando la información obtenida en un archivo json por categoría. Luego de terminado este proceso, comprime todos los archivos .json en un zip, y lo sube a una carpeta de Google Drive para luego ser procesado por Analytics.

En una primera instancia, este componente interactuaba directamente con la base de datos guardando cada publicación como un documento en una colección.

Lamentablemente, esto generaba 6 millones de documentos por día en la base de datos, lo cual hacía que ciertas operaciones como solo contar la cantidad de documentos fuera imposible de hacer. Así mismo, la extracción de estos datos era muy lenta en la práctica, y eso atenta contra la intención del equipo de brindar eficiencia a la hora de realizar los análisis, por lo cual hubo que hacer una reinvención de cómo se guardaba y procesaba la información. La solución que dio mejores resultados fue la de subir todos los archivos en un .zip, y que luego analytics guarde los datos ya procesados.

Además de procesar la información de entradas de Mercado Libre a través de la API, brinda ciertos servicios de scrapeo de publicaciones al módulo de analítica, como puede ser obtener el stock y cantidad exacta dado un código de publicación, utilizando un scraper de HTML normal en lugar de la API de Mercado Libre.

En cuanto a detalles concretos de la implementación, desde una primera instancia este componente estaba pensado hacerse en Python 3, porque al equipo le parece un lenguaje fácil para desarrollar, y ya posee experiencias scrapeando cosas utilizando librerías como BeautifulSoup.

Como se quería que el funcionamiento de este componente fuera como el de un Web Service, fue utilizado además el *framework* Flask para poder implementarlo como una API REST. Todas las interacciones con el componente, se hacen a través de diferentes *endpoints* REST (Estos pueden consultarse en el Anexo 2 - *Endpoints* de los Web *Services*).

4.4.2.2. Analytics

Este componente es el encargado de analizar la información scrapeada por Scraper. Para hacerlo, corre ciertos procesos una vez al día para actualizar la información acerca de las publicaciones y vendedores de Mercado Libre según lo scrapeado en el día.

Como se ha dicho en puntos anteriores, el atributo de calidad al cual se le ha puesto más atención en todo el desarrollo ha sido el de eficiencia, por lo cual, siempre se planeó el utilizar procesos que se ejecuten automáticamente, procesen la información,

y la inserten ya procesada en la base de datos para su posterior consulta sin tener que perder tiempo adicional en el procesamiento redundante de la información.

Este componente funciona como un *backend* a Commander. Al principio del desarrollo, el lenguaje que fue elegido había sido NodeJS con la librería Express para convertirlo en una API REST, pero se vio que la naturaleza de NodeJS con el único hilo de ejecución y el concepto de asincronicidad / promesas generaba una capa extra de dificultad al desarrollo, por lo cual se terminó usando nuevamente Python 3 con Flask para sustituir esta API de NodeJS. Con el tiempo este cambio resultó ser una buena decisión, ya que no se han vuelto a tener problemas con el desarrollo de nuevas funcionalidades, especialmente con algunos cálculos complejos que se hacían a la hora de organizar el árbol de categorías con sus nodos.

Analytics puede interactuar directamente con Scraper para consultarle acerca de información concreta de alguna publicación, e interactúa con Astroselling para el manejo del *login* y los usuarios. Este módulo no tiene información propia de usuarios, sino que toda la autenticación se hace a través de Astroselling.

Utiliza PyMongo para leer y escribir información en la base de datos.

4.4.2.3. Commander

Commander es el nombre que recibe el componente encargado de la interacción con el usuario. Es el *frontend* de la plataforma e interactúa exclusivamente con el componente de Analytics para todas sus funciones.

Se decidió hacer un componente separado solo de *frontend* para desacoplar la implementación de la lógica de la interfaz de usuario con aquella de la lógica de negocio, para así en un futuro, si así se desea, se puede crear un nuevo *frontend* para darle una “cara nueva” a la plataforma, fácilmente. Además, siendo NodeJS un lenguaje de *backend*, se necesitaba tener el *front* separado. Si bien ahora el módulo de Analytics no utiliza NodeJS sino Python, se cree que ha sido una buena decisión por la modificabilidad y flexibilidad que aporta a la interacción con el usuario final.

El lenguaje utilizado para Commander es Angular en su versión 8. Se cree que Angular ofrece todo lo que es necesario para llevar a cabo los requerimientos, aparte

de que los dos integrantes del equipo tienen experiencia con este lenguaje a pesar de no ser desarrolladores especializados en *frontend*, por lo cual fue elección del equipo frente a alternativas como por ejemplo React o VueJS. Se destaca de este tipo de lenguajes enfocados al *frontend* la facilidad para realizar el *deploy*.

En cuanto a librerías utilizadas, dada la naturaleza de Astrolytics, se hizo un especial énfasis en chartjs para poder mostrar los resultados conseguidos en distintos tipos de gráficos.

4.4.2.4. Scheduler

Scheduler no es un componente desarrollado propiamente dicho en cuanto a que no está hecho en ningún lenguaje de programación. Tanto Scraper como Astrolytics son web *services* que requieren de una interacción externa para realizar cualquier actividad, como por ejemplo comenzar el scrapeo o el procesamiento de datos. Dentro de esta rúbrica, Scheduler es el encargado de orquestar estas interacciones. Cuándo Scraper realiza el *scraping*, cuándo Analytics analiza la información, cuando genera el *ranking* de vendedores, cuando le pide a *Scraper* la información de Mercado Libre. El tiempo de cuando se ejecutan todas estas acciones es definido por Scheduler.

Al principio del desarrollo, Scheduler era un componente que estaba dentro de Astroselling, siendo varios *webjobs* que se ejecutaban a distintas horas para actualizar la información diariamente. Hoy día, es una sucesión de llamadas cURL que se ejecutan dentro del servidor de producción a diferentes horas.

A medida que el desarrollo de Astrolytics fue creciendo y se implementaron nuevas features, surgió la necesidad de ejecutar estos procesos (Procesar favoritos, procesar vendedores, procesar el *ranking*, procesar el caché de búsqueda, actualizar el árbol de categorías con respecto a Mercado Libre, etc) semanal o diariamente. La ejecución de estas llamadas cURL se hace mediante *cron jobs*.

4.4.2.5. Base de datos

La elección de la base de datos se hizo en la etapa de gestación del proyecto. Como se iba a tener una enorme cantidad de datos con diferentes atributos se consideró que lo que era necesario era una base de datos que fuera altamente modificable y

que pudiera fácilmente guardar información anidada. Por ejemplo, cuando se guardaba la información de vendedores, se guardaba para cada categoría una lista con el *top 10* de publicaciones de Mercado Libre, cada una con sus atributos, entre mucha otra información. Se consideró que esto no concuerda mucho con un modelo relacional y que para poder llevarlo a cabo en algo como esto se debía de guardar la información en formato json o utilizar muchas relaciones, por lo cual fue considerado que sería más fácil utilizar una base de datos no relacional desde el principio para guardar lo que se quería de la manera que se quisiera.

Otro de los motivos por el cual fue decidido utilizar MongoDB es por su capacidad de procesamiento y la gran cantidad de documentos que se pueden guardar. Si bien en la práctica no se pudo comprobar del todo esto porque demoraba mucho procesando 6 millones de documentos en una colección, se cree que esto se debía a que las bases de datos estaban alojadas remotamente y no local.

Actualmente, se guarda en la base de datos la información de las categorías ya procesada, lo cual es para cada categoría un documento bastante grande, pero se pudo reducir el número de entradas de 6 millones a alrededor de 12 mil, lo cual hace que sea mucho más manejable esta información. Dado el problema de que los datos obtenidos de la API de Mercado Libre para los valores de cantidad vendida y stock disponible de las publicaciones están dados dentro de un rango, no siempre se encuentra mucha diferencia entre los resultados de scrapeo de un día para el otro, por lo cual se dificulta y sea quizás redundante mantener un histórico. Por eso, todos los días la base de datos se borra y se actualiza la información insertando todo de nuevo con la información actual del día.

Se ha puesto sobre la mesa varias veces el guardar la información histórica en la base de datos, pero se cree que podría llegar a haber un problema tanto de almacenamiento como de escalamiento del procesamiento, por eso el equipo decidió dejar afuera esta funcionalidad del MVP. Sin embargo, se puede obtener información histórica de ciertas publicaciones marcadas como “favoritas” que si se guardan en la base de datos de manera histórica. Esto ahorra el problema de tener que scrapear todo Mercado Libre con información exacta (Que es imposible hacerlo en un día como se habló en el apartado de Scraper) si no que se hace exclusivamente a demanda, guardando el histórico de la información de las publicaciones favoritas de los usuarios

diariamente. Esto soluciona el problema del tiempo de scrapeado de todo Mercado Libre, y permite hacer un scrapeo mediante HTML de información que es pedida explícitamente.

Los datos en crudo quedan guardados dentro del archivo .zip de la cuenta de Google Drive para su almacenamiento y posterior consulta en caso de ser necesario, allí está el histórico de toda la información de Mercado Libre desde los primeros días de enero hasta la actualidad.

El equipo cree apropiado e interesante en este apartado mencionar el tema del caché. En el principio del proyecto, se puso sobre la mesa el agregar a los componentes un caché de Redis para poder acceder a la información procesada rápidamente, esto se haría en una primera instancia si el usuario pide la información, Analytics la procesa y luego la manda al usuario. La idea fue que luego del análisis se guarde la información en este caché para que sea intermediario en caso de que otro usuario o el mismo hiciera un pedido de la misma información, para evitar procesamiento extra. Posteriormente, se llegó a la conclusión de que esta arquitectura no necesitaba de un caché de Redis, si no que dada su naturaleza, al guardar ya la información procesada esta base de datos servía como una especie de caché.

Esta implementación también asegura ciertas cosas a la hora de que un usuario haga una consulta. La primera, es que va a responder más rápido ya que no se va a procesar la información cada vez, si no que ya está procesada. La segunda, es que se evitan errores a la hora de que el usuario pida la información, ya que en caso de que un error aconteciese, este sucedería en el proceso de análisis de datos pero no en la interacción del usuario, lo cual permite reaccionar antes y dar una mejor respuesta si sucede un error. Esto va de la mano de la eficiencia y el manejo de errores.

4.4.2.6. Astroselling

Astroselling es la plataforma del cliente, y si bien no es un desarrollo de este proyecto en particular, es un componente que interactúa en tiempo de ejecución con la plataforma.

Como se comentó en el punto de Analytics, Astroselling se encarga del manejo de usuarios, siendo quién proporciona la verificación del *login* de los usuarios e información de estos cuando intentan de iniciar sesión en Astrolytics. Esto es parte del contrato verbal con el cliente, ya que se quiere que solo los usuarios de la plataforma puedan utilizar este servicio.

Cualquier persona con una cuenta de Astroselling, sin tener que hacer nada extra, puede iniciar sesión con sus credenciales de Astroselling en el servicio y utilizarlo.

4.4.3. Interacción entre componentes.

Si bien en el apartado anterior se ha hecho una descripción de los componentes y sus interacciones, en este apartado se mostrará un diagrama de componentes y conectores y se hará más énfasis en la interacción entre estos.

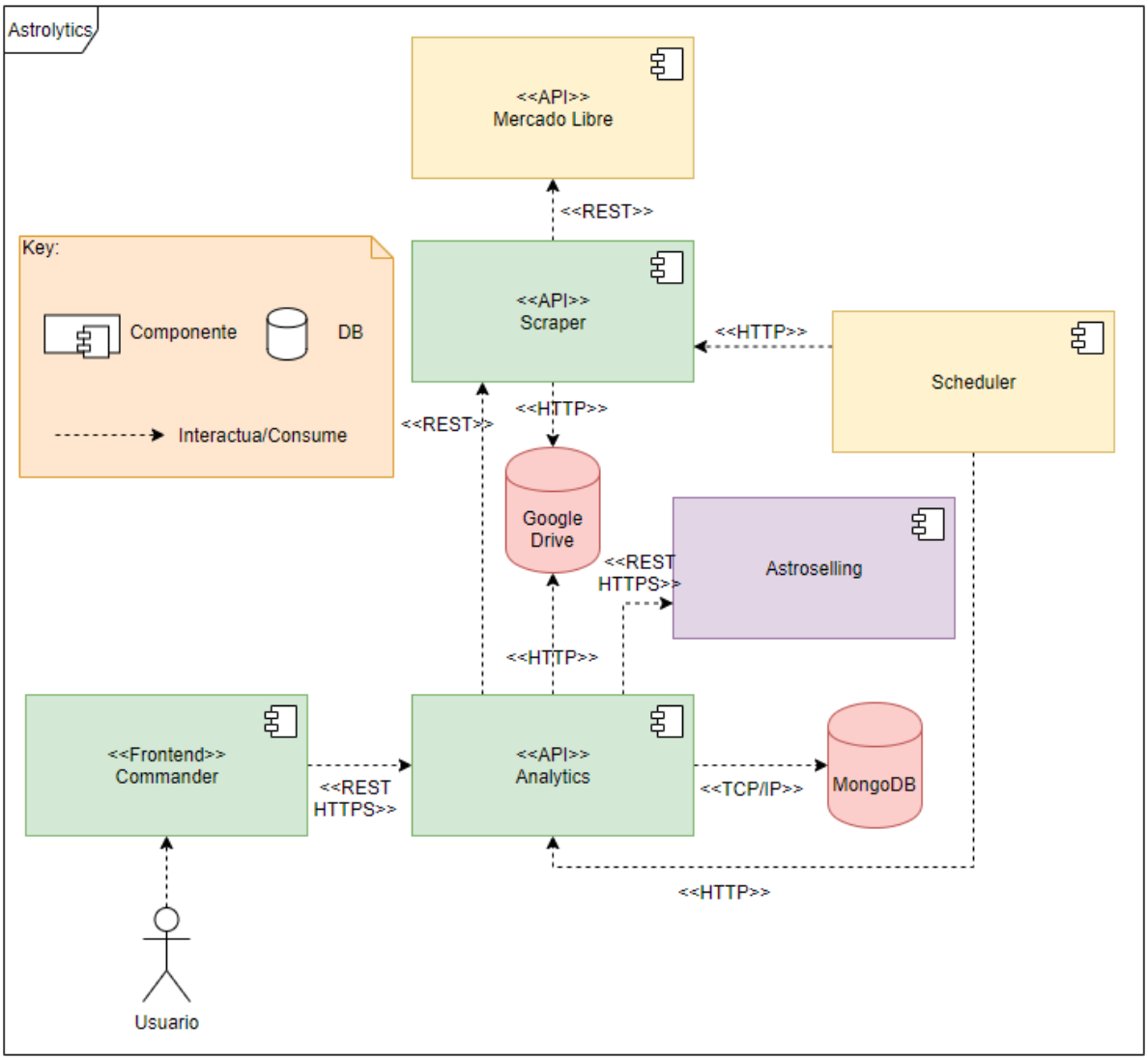


Imagen 4.4.3.1. Diagrama de componentes y conectores de Astrolytics, versión final.

En el diagrama de componentes y conectores, se ve la comunicación e interacción de las diferentes partes que componen a la plataforma.

Scraper hace llamadas a la API de Mercado Libre y a su página para realizar el scrapeo. Luego de scrapear la informaciòn, la comprime en un .zip y la sube a Google Drive.

Analytics paulatinamente toma los archivos del drive, los descarga, los descomprime y los procesa, para insertarlos en la base de datos de MongoDB. Ademàs consulta a Scraper para obtener la informaciòn exacta de los favoritos de los usuarios de

Mercado Libre, para guardar el històrico. Interactùa tambièn con Astroselling para el manejo de los usuarios y el login.

Commander recibe órdenes del usuario y consulta a Analytics para que haga pasamanos de la informaciòn que hay en la base de datos. Commander solo interactùa con Analytics y no con ninguno de los otros componentes de la soluciòn.

Scheduler le da òrdenes mediante cURL automàticamente a Analytics y a Scraper para que ejecuten sus procesos diarios.

4.4.4. *Deployment*

4.4.4.1. Ambientes y *hosting*

En los inicios del proyecto el *deploy* se hacía sobre el servicio en la nube de IBM, que fue proporcionado gratuitamente por el cliente Astroselling. Ya desde una etapa temprana del proyecto era importante para el equipo poder tener la plataforma en la nube o por lo menos el componente Scraper para que pudiera scrapear la informaciòn automàticamente sin tener que ejecutarlo en las computadoras del equipo de desarrollo.

Bajo este servicio, se hizo el *deploy* de los tres componentes principales Scraper*,* Analytics *y* Commander. Para hacer el *deploy* de los primeros dos se tomó una plantilla de *server* de Python de IBM, y para este último solo se copiaron y pegaron los archivos transpilados de typescript como es pràctica habitual en estos lenguajes. El componente de Scheduler era una serie de *webjobs* corriendo sobre la plataforma Azure de Microsoft.

Astroselling tenìa algunos componentes de su soluciòn tanto en IBM como en Azure. Con el paso del tiempo, decidieron que les era estratègicamente mejor tener todo centralizado y no estaban del todo conformes con ciertas cosas de Azure, por lo cual migraron a Amazon.

Para poder tener cierta independencia en el desarrollo y en el resto de los procesos, se decidió hostearlo independientemente de Astroselling por la duraciòn de la tesis, para evitar posibles migraciones o tener que consultar por cada componente nuevo o cambio a estos. Ademàs, se encontró una diferencia econòmica significativa con respecto a tener que levantar tantos componentes de manera separada.

Es por este motivo que se decidiò usar la plataforma Digital Ocean para comprar acceso a un servidor con Ubuntu, denominado *droplet*. Este servidor sirvió para hostear los tres componentes principales y Scheduler, utilizando Apache 2 y el *crontab* de linux respectivamente.

La base de datos está alojada en uno de los servicios de *database as a service* de Atlas, propiedad de MongoDB, y asì ha sido desde el principio del desarrollo, incluso

antes del *deploy*. Se piensa que lo mejor era tener una base de datos unificada en todo momento por los datos y el volumen de estos que es manejado.

En cuanto a distintos ambientes, no se posee ambiente de *staging* ni de desarrollo, si no que se cuenta unìcamente con el denominado “producciòn”. El equipo considera que no era necesaria dada la naturaleza del producto que se apuntaba a obtener. Por ejemplo, no parecía viable tener una base de datos diferente para cada ambiente por la redundancia en el procesamiento de datos que ya de por sì es costosa en tiempo y recursos para el servidor.

Se considera que no era conveniente tener màs *droplets* porque el costo que agregaba no es solo monetario, sino también de esfuerzo de desarrollo y de pruebas los cuales no aportaba nada relevante a lo que era el MVP.

4.4.4.2. Dominios

Desde el primer momento se consideró que Astrolytics iba a ser un mòdulo de Astroselling, por lo cual, segùn lo hablado con el cliente se iba a poder acceder a un subdominio de Astroselling para usar la plataforma.

Sin embargo, hubo algunos atrasos burocráticos para cuando la plataforma estaba lista para ser liberada, por lo cual se creò momentàneamente el dominio de astrolytics.info, haciendo que el dominio principal fuera para Commander y subdominios de scrapper.astrolytics.info y api.astrolytics.info para Scraper y Analytics respectivamente.

Con el paso del tiempo, los impedimentos burocráticos pasaron, y el equipo pudo acceder al subdominio stats.astroselling.com para el mòdulo de Commander, dejando los otros dos dominios como estaban.

Se cree oportuno tambièn que desde que se tienen dominios separados para cada aplicaciòn, las aplicaciones corren en puertos estàndar como 80/443. Cuando no se tenìan los dominios cada aplicaciòn era accedida mediante la IP y luego el puerto. Teniendo Analytics el puerto 5000, Scraper el puerto 5001, y Analytics el 80.

Uno de los pasos que se tuvo que hacer para poder subirlo a producciòn y liberar la versiòn “beta” del MVP para que usen ya usuarios finales, fue tener que implementar

https en los servidores. Esto se hizo a travès del Certbot de Let 's Encrypt, ejecutandolo sobre el *droplet* de Digital Ocean.

Se logró aplicar el certificado sobre los 3 dominios mencionados anteriormente, con el redireccionamiento correspondiente. El certificado tiene una duraciòn de 3 meses y se renueva automàticamente.

4.5. Construcción de versiones

No se tiene un estricto control de versionado para ninguna de las aplicaciones, pero en líneas generales dado que se manejaron en un desarrollo incremental guiado por sprint, diría que los cambios hechos en cada sprint son un incremento en la versión, y la versión sería el número del sprint.

Existe un hito importante en el desarrollo del programa, que es cuando fue subido a producción para que lo puedan utilizar clientes de Astroselling. Al equipo le parece apropiado llamar a esta versión la versión 1.x. Todas las versiones anteriores son versiones 0.x, donde x es el número de *sprint* actual en los dos casos. Se prefiere que el versionado sea homogéneo para todos los componentes en lugar de separarlos, ya que a pesar de estar separados, son vistos como una plataforma sola. Por eso, en lugar de hacer que cada componente tenga una versión por separada, es preferible darle versionado a la plataforma en general de Astrolytics. La última versión del MVP es la 1.12.4.

En la tabla a continuación se muestran algunos de los registros del versionado interno.

Versión	Fecha	Funcionalidad
0.1	30/12/2019	Creación del proyecto fuera del *Proof of concept.*
0.2	13/01/2020	*Deploy* a IBM, *Login* con Astroselling.
0.3	27/01/2020	Cambio de NodeJS a Python módulo de analítica, *Frontend*.
0.4	10/02/2020	Explorar Mercado Libre, mis Ventas.
0.5	24/02/2020	Explorar Mercado Libre, parte 2.
0.6	09/03/2020	Reparación de *bugs*, se comenzó el módulo de vendedores.
0.7	23/03/2020	Reparación de *bugs*, comparación de tiendas, información de tiendas. Favoritos.
0.8	06/04/2020	*Dashboard* con publicaciones que estoy siguiendo, Página de búsqueda de vendedores. *Ranking* de vendedores.
0.9	20/04/2020	Reparación de *bugs*, gráfico de evolución de precios.

1	04/05/2020	Pasaje a producción en Digital Ocean.
1.11	18/05/2020	Búsqueda de items. Formulario de *feedback*.
1.12	01/06/2020	Mejoras a búsqueda de items, inicio de recomendaciones.
1.12.1	15/06/2020	Recomendaciones
1.12.2	29/06/2020	*Revenue* diario para las empresas.
1.12.3		*Revenue* diario para las empresas.
1.12.4		Correcciones generales.

Tabla 4.5.0.1. Versionado de Astrolytics

4.5.1. Paquetes y módulos implementados

Cada uno de los tres componentes es su propia aplicación, con sus propias dependencias y código fuente. Como se comentó en los puntos anteriores, Scraper y Analytics están hechos en Python, mientras que Commander está desarrollado en Angular.

Para los dos primeros, no es necesario compilar para hacer el *deploy* en producción, si no que solo basta con utilizar WSGI configurado con Apache para que se puedan ejecutar con los archivos .py sin compilar.

En el caso de Commander, se utiliza el comando *"ng build"* que trae la plataforma ng para hacer la transpilación de código de typescript a javascript, teniendo como resultado una carpeta dist con los js necesario para que el navegador del usuario pueda ejecutar la aplicación localmente.

4.5.2. Herramientas y ambientes utilizados para desarrollo

Para el desarrollo del código, fue utilizado el entorno de desarrollo integrado Visual Studio Code para los tres componentes. Ya que es el que es utilizado habitualmente por el equipo en su vida diaria, y los lenguajes de programación que son utilizados no requieren herramientas adicionales para la compilación como puede ser una aplicación de .NET o de Java.

Para las aplicaciones en Python, se utilizó el plugin de Python de visual studio para IntelliSense, *linter* y formato de código. Para poder levantar los ambientes se utilizó la línea de comandos y Virtual Env para no tener problemas con las versiones locales ni las dependencias de Python. Para el control de dependencias se utilizó PIP

Para Commander, se utilizó "*ng serve*" para el ambiente local de desarrollo, y "*ng build*" para realizar la *build*. Para el control de dependencias se utilizó npm.

4.6. Aseguramiento de calidad y plan de calidad

En este capítulo se describirán las consideraciones que se tuvieron en cuenta al momento de diseñar e implementar un proceso de SQA apropiado para el desarrollo de Astrolytics.

4.6.1. Descripcion y justificacion del proceso de SQA

Para el diseño de un proceso de Software Quality Assurance (SQA) apropiado se tuvo que considerar el contexto del desarrollo. En “Handbook of Software Quality Assurance” se definen los conceptos de “Organización Pequeña”[17] y “Proyecto Pequeño”, los cuales consisten en organizaciones y proyectos que cuentan con menos de 150 y 25 personas, respectivamente. Astrolytics al ser un producto desarrollado por dos estudiantes de ingeniería cae dentro de estas categorías, lo cual debe ser tomado en cuenta al momento de desarrollar el proceso SQA.

El detalle con el que se debe contar es los recursos limitados del equipo. Al ser solamente dos integrantes, ninguno de los dos un ingeniero de calidad entrenado, es importante elegir actividades para conformar un proceso de SQA de manera que que no sea detrimental al desarrollo de la aplicación en sí. En lo posible, que sean fácilmente acoplables al proceso, que presenten una sinergia y aporten al desarrollo.

El equipo planteó la definición de calidad propuesta en “Handbook of Software Quality Assurance”, donde la calidad del producto se relaciona directamente con la cantidad de requerimientos que el mismo satisface. [17]

4.6.2. Detalle de actividades

Antes de comenzar el desarrollo, el equipo tuvo una reunión donde se discutió sobre estándares de codificación y buenas prácticas a utilizar. Esto se realizó con tal de homogeneizar el código entre ambos integrantes y aumentar la legibilidad del mismo, disminuyendo así el tiempo perdido que pueda implicar leer código entre integrantes y mejorando la mantenibilidad, teniendo en cuenta la posterior cesión de derechos de Astrolytics.

Luego, se integró al proceso de desarrollo de nuevas funcionalidades el método de pull requests revisadas. Al utilizar la metodología Gitflow[18], explicada en detalle más adelante, el desarrollo se administra en ramas de desarrollo paralelas, las cuales son eventualmente unidas a la rama maestra. Para la realización de este proceso de unión, se utiliza un evento llamado "*Pull Request*", donde la unión de las ramas es pausada hasta ser auditada por una cantidad de miembros del equipo. En este caso, el otro miembro del equipo que no desarrolló la funcionalidad debe revisar los cambios en el código, con la intención de encontrar detalles a corregir. Estos detalles pueden ser de índole nomenclatura, *clean code*, oportunidades de refactorización apuntando a eficiencia, etc.

Tomando en cuenta que gran parte de la calidad del producto radica en la satisfacción de las necesidades del cliente [17], y también que un aspecto del desarrollo es que no se cuenta con requisitos bien definidos, se consideró de crítica importancia la correcta inferencia y definición de los requerimientos, junto con la validacion y verificacion apropiada de las funcionalidades desarrolladas. Para cubrir este aspecto de calidad, se tuvo un énfasis importante en la comunicación con el cliente y los usuarios de Astrolytics, organizando reuniones de manera periódica para demostrar el progreso de la aplicación y obtener *feedback* para incorporar al desarrollo.

Por otro lado, también se prestó atención al análisis de los atributos de calidad involucrados con el desarrollo. Se consideró que es importante tenerlos en cuenta al momento de inferir requerimientos y desarrollar, ya que estos atributos son reflejados en la arquitectura del sistema.

En cuanto a las pruebas para el aseguramiento de la calidad, en una primera instancia no se fueron consideradas formalmente, ya que por la naturaleza de MVP del proyecto y que es un proceso iterativo que se retroalimenta del feedback de los usuarios, iba a haber un movimiento y un testeo constante de la plataforma en cuanto a la funcionalidades. Adicionalmente, Astrolytics utiliza varios servicios externos como Astroselling y Mercado Libre, por lo cual se hace un poco más difícil el proceso de pruebas.

A pesar de esto, luego de considerar que hay una parte de la funcionalidad que es muy importante que funcione correctamente y depende de ciertos cálculos, se

concluyo que es necesario un set de pruebas formal que cerciore y garantice el correcto funcionamiento y generación de los datos a Astrosellng. Por lo tanto, se harán un *set* de pruebas que verifiquen que el "núcleo" del programa funcione correctamente.

Tomando en cuenta la usabilidad del sistema, el equipo compuso también un manual de usuario. El manual se ofrece en la sección de preguntas frecuentes de Astrolytics. Este es actualizado con cada funcionalidad nueva, y fue estilizado siguiendo una guía de uso de marca proporcionada por Astroselling, con la intención de homogeneizar el estilo de los entregables. El manual se puede encontrar en el Anexo 4.

4.6.3. Productos resultantes

Entre los productos resultantes de los procesos de SQA se pueden encontrar los siguientes elementos:

- *Backlog* de tareas
- Detalle de entrevistas realizadas (Anexo 3)
- Seguimiento cronológico del trabajo realizado y horas invertidas semanalmente
- Prototipos y versiones de Astrolytics utilizadas para verificación
- *Sets* de pruebas con sus resultados.
- Métricas del producto

4.6.4. Métricas del producto y proceso

4.6.4.1. Horas totales de trabajo realizado

A lo largo del proyecto, el equipo registró sus horas de trabajo en el proyecto con la ayuda de la herramienta Toggl. De esta manera se logró obtener un seguimiento del trabajo invertido por miembro y tarea relacionada al proyecto. Esto no solo sirvió a nivel de equipo para obtener las métricas, sino que fue útil a nivel individual como incentivo de trabajo y referencia de autoevaluación.

Como fue mencionado anteriormente, el proyecto se dividió en tres etapas, la investigación inicial, el desarrollo principal y la finalización del proyecto. De estas tres etapas solamente el desarrollo principal tuvo una gestión del tiempo invertido formal con el uso de la herramienta. A partir de los registros se pudo obtener un resultado de 943,25 horas invertidas en esta etapa, las cuales están divididas de manera heterogénea en 14 sprints de desarrollo, cada uno de dos semanas de extensión.

En la primera etapa de investigación inicial y desarrollo del *proof of concept* no se realizó un seguimiento formal de las horas invertidas en el proyecto.

En los dos últimos meses de desarrollo, los cuales involucraron más aspectos de soporte y documentación de la solución, tampoco se realizó un registro formal del tiempo invertido. Esto se debe al hecho de que se decidió no utilizar más el formato *Scrum* para el desarrollo de la aplicación. Sin embargo, el hábito de invertir 20 horas semanales al proyecto por miembro permaneció hasta el fin del desarrollo.

4.6.4.2. Puntos de trabajo por *sprint*

La siguiente métrica que fue utilizada para realizar un seguimiento del proceso de desarrollo consiste en los puntos de trabajo completados por *sprint*. Esta métrica permitió al equipo inferir su capacidad de trabajo por *sprint*, y consecuentemente ajustar su *backlog* de trabajo. De esta manera el equipo pudo establecer un ritmo constante de trabajo que les permitió estimar correctamente el tiempo requerido para realizar incrementos en la solución.

4.6.4.3. Puntos de trabajo por hora por *sprint*

La última métrica relacionada al seguimiento temporal del proyecto consiste en un seguimiento de los puntos de trabajo completados y su relación a las horas invertidas en el *sprint*. Se le denomino eficiencia del equipo a nivel interno, y es una métrica que indica precisamente eso, cómo se traducen las horas invertidas del equipo a trabajo realizado. Cumple una función de autoevaluación para los integrantes y sirve como objetivo, dado que el equipo apunta a maximizar esta eficiencia, repitiendo aquellos procesos o actividades que hayan sido útiles en el *sprint*.

El seguimiento de estas métricas fue realizado en una tablilla interna al equipo, mostrada a continuación.

#	Inicio	Fin	Puntos Done	Puntos Doing	Puntos To Do	Horas de trabajo totales	Puntos por hora
0	16/12/19	30/12/19	26	5	12	27,75	0,94
1	30/12/19	13/1/20	30	11	0	43	0,70
2	14/1/19	27/1/20	32	8	5	46	0,70
3	27/01/20	10/2/20	65	10	16	74,75	0,87
4	11/2/20	25/2/20	78	13	5	74	1,05
5	26/2/20	09/03/2020	73	38	5	82	0,89
6	10/03/2020	23/03/2020	64	16	1	75,25	0,85
7	23/03/2020	06/04/2020	65	30	0	88	0,74
8	07/04/2020	20/04/2020	78	8	21	79	0,99
9	20/04/2020	04/05/2020	89	0	8	75,5	1,18
10	05/05/2020	18/05/2020	32	42		64	0,50
11	19/05/2020	01/06/2020	88	14		70	1,26
12	02/06/2020	15/06/2020	80	0	0	77	1,04
13	16/06/2020	29/06/2020	38	33	9	67	0,57

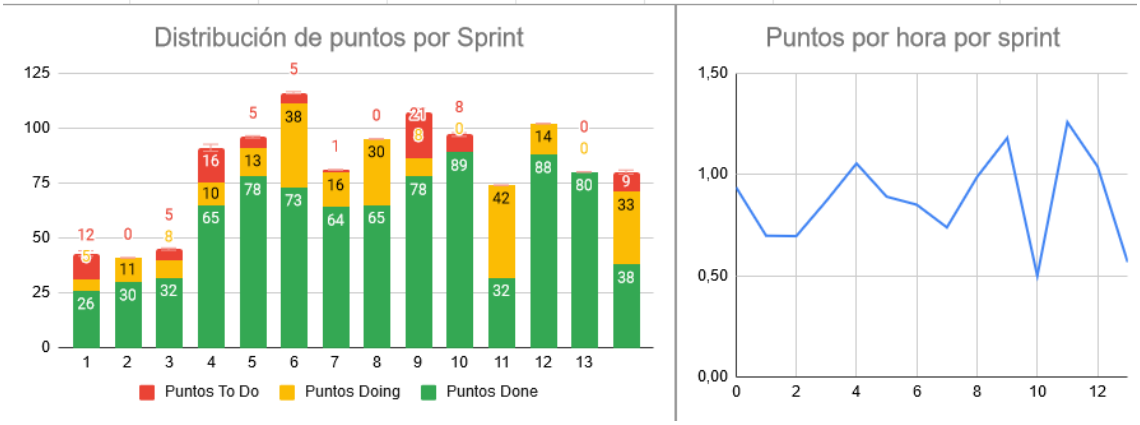


Imagen 4.6.4.3.1. Registro y graficaciones de métricas de trabajo

Junto con los registros, el equipo también grafico una distribución del estado final de los puntos por *sprint*, y una evolución de la métrica de puntos por hora por *sprint* a lo largo de los *sprints* del proyecto. A partir de estos registros, el equipo pudo concluir

que la carga ideal de puntos para el *sprint* debe encontrarse a prueba y error, gradualmente incrementando la carga y evaluando los resultados. También fue posible contemplar la “Ley de Parkinson”[19] en acción, la cual enuncia “*El trabajo se expande para llenar el tiempo disponible para su finalización*”. Esto se debe a que en los *sprints* de 40 puntos el trabajo no fue 100% completado, a pesar de que en los *sprints* de 80 puntos, se realizaban más de 40 puntos cada *sprint*.

Los puntos por hora por *sprint* (eficiencia del equipo) fueron útiles a un nivel más granular del equipo. Esta métrica permitió a cada integrante evaluar su eficiencia y capacidad de trabajo, brindando la posibilidad de evaluar métodos de trabajo distintos y ajustarlos en cada *sprint*, ajustandolos acordemente en el siguiente.

4.6.4.4. Pruebas

El análisis de datos, que es la parte más importante y compleja de Astrolytics en cuanto a cálculos, funciona de la siguiente manera: Analytics descarga el último archivo de resultados de Scraper, lo descomprime, analiza todos los archivos y luego los resultados los sube en la base de datos. La acción que gatilla este procesamiento de datos es una llamada a un *endpoint* dentro de Analytics, y funciona como un pipeline donde los datos se procesan secuencialmente.

Este comportamiento de *pipeline* secuencial dificulta notoriamente el *testeo* de las funcionalidades por separado por cómo está construido el programa (especialmente porque los pasos posteriores se alimentan de los resultados de sus pasos anteriores). Por lo cual se decidió hacer un enfoque de caja negra a las pruebas, modificando la entrada y viendo si la salida al final del *pipeline* coincidía con la esperada.

Para poder hacer eso, se crearon archivos *mock* que son los archivos que procesa Scraper, y a partir de saber lo que contiene el archivo, se pueden hacer los cálculos a mano para obtener los resultados esperados, y compararlos con el resto. Como los resultados de la función son grabados directamente en la base de datos, se creó una base de prueba para realizarlas.

Se hicieron en total 22 pruebas en 2 funciones.

```
Testing generateListingRanking_withoutFiles...OK (1/22)
Testing generateListingRanking_oneFileOneListing_ranking...OK (2/22)
Testing generateListingRanking_oneFileOneListing_graphs...OK (3/22)
Testing generateListingRanking_oneFileTwoListingsSameSeller_ranking...OK (4/22)
Testing generateListingRanking_oneFileTwoListingsSameSeller_graphs...OK (5/22)
Testing generateListingRanking_oneFileTwoListingsDifferentSellers_ranking...OK (6/22)
Testing generateListingRanking_oneFileTwoListingsDifferentSellers_graphs...OK (7/22)
Testing generateListingRanking_twoFilesOneListingSameSeller_ranking...OK (8/22)
Testing generateListingRanking_twoFilesOneListingSameSeller_graphs...OK (9/22)
Testing generateListingRanking_oneFileTenListingsTenSellers_graphs...OK (10/22)
Testing generateListingRanking_oneFileElevenListingsElevenSellers_graphs...OK (11/22)
Testing generateListingRanking_oneFileTwelveListingsTwelveSellers_graphs...OK (12/22)
Testing generateListingRanking_threeFilesOneListingNestedCategories_graphs...OK (13/22)
Testing generateSellersData_withoutFiles...OK (14/22)
Testing generateSellersData_oneFileOneListingUnderThreshold...OK (15/22)
Testing generateSellersData_oneFileOneListingOverThreshold...OK (16/22)
Testing generateSellersData_oneFileTwoListingOverThreshold...OK (17/22)
Testing generateSellersData_oneFileTwoListingDifferentSellersOneNotOverThresholdAndOtherOverThreshold...OK (18/22)
Testing generateSellersData_twoFilesTwoListingsOverThreshold...OK (19/22)
Testing generateSellersData_tenFilesTenListingsOverThreshold...OK (20/22)
Testing generateSellersData_fifteenFilesFifteenListingsOverThreshold...OK (21/22)
Testing generateSellersData_sixteenFilesSixteenListingsOverThreshold...OK (22/22)
```

Imagen 4.6.4.4.1. Resultado de las pruebas automatizadas

En la tabla a continuación se resumen algunas métricas de las pruebas.

Se define en este caso a un problema como algo que no estaba dentro de los resultados esperados, pero que no tiene un impacto mayor en el software. Por ejemplo, por un tema de redondeo hay una diferencia de un 0.1% en un total calculado.

Un error se define como una diferencia detectada entre lo obtenido y esperado que afecta la calidad de los resultados del programa y hay que corregirlo.

Función	Pruebas	Problemas detectados	Errores detectados
generateListingRanking	13	2	0
generateSellersData	9	0	0

Tabla 4.6.4.4.2. Mètricas de las pruebas.

A la hora de desarrollar estas funciones se habían hecho unas pruebas para garantizar su funcionamiento pero no se habían formalizado como se hizo en este caso. En esta primera instancia anterior informal se habían detectado errores que se corrigieron, eso puede explicar la falta de errores detectados en esta instancia.

4.6.4.5 Métricas de *Performance*

Para las pruebas de *performance*, se utilizó el programa JMeter con algunos *endpoints* de la solución que se consideran representativos.

El primer *endpoint* elegido es el que se usa para obtener los favoritos del usuario, que consta de una única consulta a la base de datos en base a una *query* del id del usuario. El segundo *endpoint* devuelve la información de las cuatro gráficas y la tabla del *dashboard*, que es un volumen medianamente grande de información, llamado Dashboard Items. El tercer *endpoint* es uno que hace algo análogo pero para el *revenue* diario de cada tienda, que lleva una carga de procesamiento extra lo cual lo hace el *endpoint* más “pesado” en cuanto a procesamiento. El cuarto *endpoint* es la información de una categoría de Mercado Libre, y el quinto y último *endpoint* analizado es el *endpoint* que devuelve la información de la tienda con más ventas de Mercado Libre Uruguay.

Las especificaciones de las pruebas fueron utilizando un *ramp-up period* de 60 segundos para todos los casos, y un *loop count* de 10, por lo cual cada prueba se repite una diez veces. El dato obtenido de estas métricas es el tiempo promedio de respuesta en milisegundos (ms) para cada uno de los *endpoints* para diferentes cantidades de *threads* (usuarios concurrentes). Se tomaron métricas para 5, 10, 20, 50 y 100 usuarios concurrentes con un total de 50, 100, 200, 500 y 1000 requests de diagnóstico por *endpoint* respectivamente.

Como se discutió en el apartado de atributos de calidad, dado que es un MVP se espera en el peor de los casos 10 usuarios concurrentes usando la plataforma, y se considera para la gran mayoría de los casos como aceptable un tiempo de respuesta inferior a 2000 ms.

En la tabla a continuación se puede observar el tiempo de respuesta obtenido en milisegundos para cada *endpoint*/cantidad de usuarios concurrentes.

Endpoint	5 Usuarios	10 Usuarios	20 Usuarios	50 Usuarios	100 Usuarios
User favorites	773 ms	767 ms	740 ms	820 ms	842 ms
Dashboard Items	675 ms	577 ms	588 ms	656 ms	2588 ms
Dashboard Stores	2170 ms	5797 ms	12695 ms	-	-
MLU1055 Statistics	336 ms	327 ms	327 ms	345 ms	1282 ms
Seller 64834284 Statistics	589 ms	566 ms	553 ms	646 ms	1743 ms

Tabla 4.6.4.5.1. Métricas de *performance*

Se pueden observar tiempos de respuesta satisfactorios para cuatro de los cinco *endpoints* definidos, con la excepción del *endpoint* para obtener la información de *revenue* diario de las tiendas. Se dejaron de hacer pruebas luego de 20 usuarios concurrentes para este caso porque era inviable realizarlas.

A partir de este análisis, se sugieren algunas formas de mejorar el tiempo de respuesta concreto para este caso para cumplir satisfactoriamente con nuestros requisitos de *performance*. Cabe destacar que, en la práctica, el *frontend* guarda en el caché del navegador por un tiempo definido por el proyecto (Actualmente son 6 horas) la respuesta de este *endpoint* y el de Dashboard Items, lo cual hace que solo se haga la consulta al *backend* una vez cada seis horas por usuario. Así mismo, la llamada hecha al *backend* por este *endpoint* se realiza después de que se carga la página asincrónicamente, por lo cual el usuario puede acceder a otra información mientras espera respuesta del *backend* para mostrar esta gráfica. Mientras tanto, el usuario ve un ícono de una tuerca girando, pero permite realizar el resto de las acciones. Esta es la forma que se encontró en el corto plazo para minimizar el impacto de esta demora.

Con este punto explicado, se nos ocurren otras formas para mejorar la *performance* de este *endpoint* en particular. Como primer punto, optimizar el algoritmo para que haga menos operaciones en la base de datos. De segunda sugerencia, utilizar un caché para que los cálculos y los resultados del análisis de los datos sean guardados temporalmente, de este modo si se hace una consulta sobre una tienda que ya está almacenada en el caché no tenga que calcularse ni irse a buscar en la base de datos nuevamente. Esta segunda opción no se hizo porque no se consideró necesario

agregar la complejidad de un servidor de Redis al servidor en la nube para el MVP, si no que se implementó el caché del lado del cliente. Como tercera opción, se puede mejorar las especificaciones del servidor que se está usando, o utilizar un servidor de MongoDB local o con menos latencia con respecto a la máquina virtual. Como se vio en los apartados anteriores, existe una latencia intrínseca e inexorable entre el *backend* y la base de datos ya que están física y virtualmente en lugares distintos.

La *performance* de todas las operaciones mejoraría significativamente (especialmente la de Dashboard Stores) si se hiciese una instalación de MongoDB localmente en donde está hosteado el servidor.

Se graficaron algunos de los datos obtenidos para poder visualizar mejor la diferencia de velocidad de respuesta en milisegundos en función de la cantidad de usuarios concurrentes para cada uno de los *endpoints*, salvo el *endpoint* del cual se habló anteriormente.

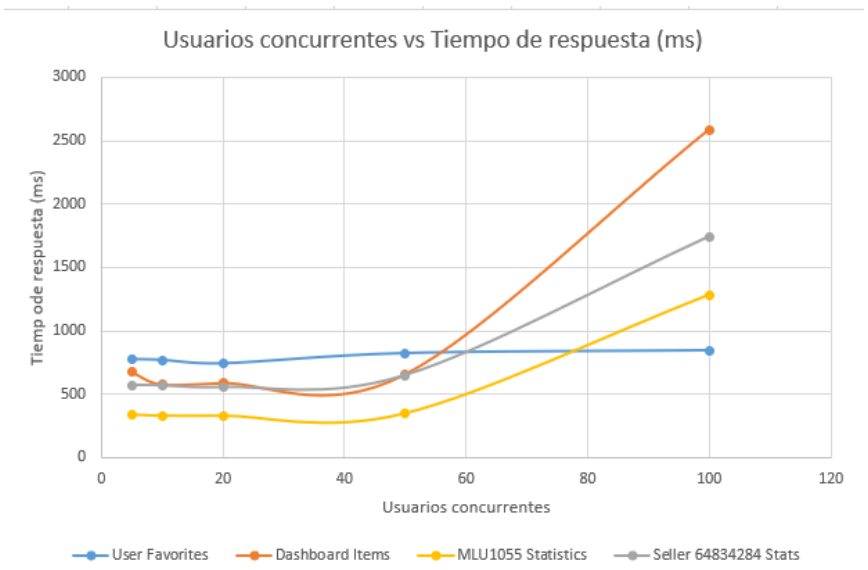


Imagen 4.6.4.5.2. Tiempo de respuesta en función de usuarios concurrentes

Se puede ver que no hay mucha diferencia hasta 50 usuarios concurrentes y que cada uno de los tiempos de respuesta para este intervalo son inferiores a 1000 ms, notándose un aumento significativo de los tiempos de respuesta para 100 usuarios concurrentes para cada caso.

Dados estos resultados los tiempos de respuesta se consideran satisfactorios en su gran mayoría para cumplir con nuestro atributo de calidad de *performance* para 10 usuarios concurrentes, que es el peor caso posible proyectado para el alcance actual del proyecto. Para el caso no satisfactorio, se presentaron alternativas y soluciones para hacer en etapas posteriores no contempladas en este proyecto.

4.7. Gestión de la configuración

4.7.1. Elementos de configuración

4.7.1.1. Control de la configuraciòn

Los parámetros de configuración que requieren un nivel de seguridad mayor como lo son las credenciales de acceso a las bases de datos o secretos para la intercomunicación entre los servicios no se guardará en el repositorio, sino que se pondrán como parámetros de configuración de la aplicación dentro del proveedor de servicios para cada una de las *web apps* cuando la plataforma sea puesta en producción.

Existirán parámetros de configuración predeterminados para todos los campos en todos los componentes, que funcionarán de manera predeterminada en caso que no haya variables de entorno configuradas en el *deploy*. Estos parámetros corresponden a los de prueba y se piensa en un futuro poder reemplazarlos con los de producción una vez estos se *setteen* como variables de ambiente.

4.7.1.2. Integración contínua

Al empezar el *deploy* de proyecto en la plataforma de IBM, se configuró la utilización de un proceso de integración contínua, de modo que cada vez que se haga un *push* a la rama *master*, se copie el código desde el repositorio al ambiente, y se propaguen los cambios para poder verlos.

Con el traspaso a Digital Ocean, al día de escribir esto no se ha realizado la integración contínua para este nuevo ambiente, ya que si bien es considerado un ambiente que va a durar hasta que termine el proyecto en lo académico, es un *host* temporal de la plataforma. Por cómo se tienen organizados los cambios en base a los *sprints*, se hacen actualizaciones a los componentes una vez cada dos semanas, por lo cual se considera que de momento no es necesario aplicar integración contínua.

4.7.2. Repositorio

4.7.2.1. Control de versiones

Para guardar el código y hacer un seguimiento de este, se utilizará git hosteado en la plataforma Github, donde cada componente tendrá su propio repositorio.

Se utilizará la metodología de gitflow[18] para hacer el trabajo diario dentro del repositorio, teniéndose una rama para desarrollo y otra para producción. Cada vez que haya una *feature* nueva, se hará una *branch* con el nombre de la *feature*, y se hará un procedimiento análogo para los *fixes* y *hotfixes*.

El programa predilecto usado por el equipo es SourceTree de la empresa Atlassian.

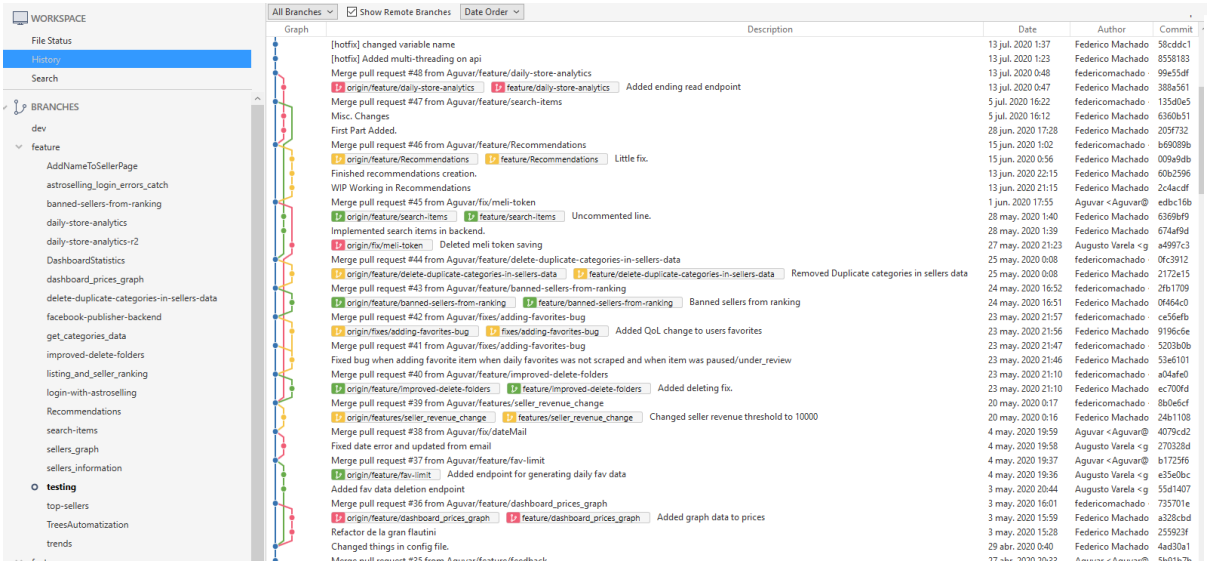


Imagen 4.7.2.1.1. Repositorio de Analytics visualizado en SourceTree

5. Conclusiones

5.1. Conclusiones generales

Al comienzo del proyecto el equipo se propuso metas y objetivos a cumplir como criterio para declarar al desarrollo como exitoso. Se considera que se ha logrado cumplir los mismos, y también se han logrado obtener lecciones importantes a tener en cuenta para próximos proyectos. Estas lecciones pueden interpretarse como buenas prácticas a seguir e implementar, o como situaciones o acciones a evitar.

La duración del proyecto constó de 12 meses, dividido en tres etapas de desarrollo. Durante la etapa de desarrollo principal desde el 16 de diciembre del 2019 hasta el 29 de junio del 2020 se registraron un total de 943,25 horas de trabajo por parte de los dos integrantes del equipo.

Las horas invertidas en las etapas de investigación inicial y soporte final de la aplicación no fueron registradas, por lo que no se cuenta con un número concreto referente a las horas invertidas en ellas.

5.2. Resumen de los productos resultantes en función de los objetivos trazados

Con respecto a los objetivos del proyecto, se logró desarrollar un MVP usable, que fue puesto en un ambiente de producción. Este se encarga de scrapear información de Mercado Libre Uruguay diariamente, procesando sus datos y ofreciéndole a sus usuarios métricas y detalles útiles para decisiones estratégicas.

El cliente se encuentra conforme con el desarrollo realizado. Se les entregó una encuesta para medir su nivel de satisfacción, la cual puede ser encontrada en la siguiente sección. Actualmente se encuentran prestando el servicio de Astrolytics a más de sus clientes conforme ellos comunican que encuentran el servicio util y comodo de utilizar. Han decidido conservar el producto, continuando su desarrollo posterior a la finalización del proyecto.

A nivel académico, el equipo realizó un desarrollo de largo plazo, gestionando sus aspectos temporales, backlog, inferencia de requerimientos y programación. Se trabajaron con tecnologías nuevas y conocidas, se realizó un *deploy* y un monitoreo de 3 aplicaciones conviviendo en un ecosistema, y también se administró soporte y mantenimiento por un periodo a la aplicación. Se considera que los objetivos académicos fueron cumplidos, logrando gestionar un proyecto de desarrollo de software de largo plazo satisfactoriamente.

5.3. Encuesta de satisfacción del cliente

Al terminar formalmente el desarrollo de la solución, se decidió hacer una encuesta de satisfacción a Astroselling, para ver si estaban conformes o no con el rendimiento del equipo, qué pensaban del funcionamiento de la plataforma y si tienen sugerencias nuevas a futuro.

Las preguntas que se realizaron, alineadas con nuestros objetivos, fueron las siguientes:

- ¿Cómo considera el trato personal que ha tenido el equipo de Astrolytics con el equipo de Astroselling?
- ¿Qué tanto siente usted que se ha cumplido con el objetivo propuesto desde un principio, con respecto al producto desarrollado?
- ¿Considera que el equipo de Astrolytics los ha mantenido al tanto de las nuevas funcionalidades a medida que se ha avanzado en el desarrollo y ha sido receptivo en cuanto a sus comentarios?
- ¿Qué tanto piensa que el equipo de Astrolytics ha logrado entender las necesidades de Astroselling y de sus usuarios a la hora de desarrollar nuevas funcionalidades?
- A la hora de haber probado la plataforma. ¿Le parece que ha sido estable y sin errores inesperados en su uso?
- ¿Considera que la plataforma es navegable de forma rápida?
- ¿Le parece fácil de usar y de entender en cuanto a las funcionalidades que ofrece y cómo visualizarlas?
- ¿Qué tan útil le parece la herramienta desarrollada?
- ¿Ha notado interés en parte de los clientes de Astroselling para el uso de la herramienta?

Las preguntas recién descritas tenían un rango de evaluación del 1 al 5. Esto se hizo primero que nada para poder normalizar los resultados, y en una segunda instancia para no abrumar a quien conteste el cuestionario

Posteriormente, se preguntó si se consideraba que Astrolytics ayuda a darle más valor al usuario de Astroselling, y dos preguntas donde los encuestados podían explayarse

en más profundidad si así lo preferían: ¿Qué funcionalidades les gustaría ver en un futuro? y ¿Tiene algún otro comentario acerca de Astrolytics? Las dos opcionales.

Los resultados de esta encuesta se pueden encontrar en el Anexo 7 - Resultado de encuestas de satisfacción al cliente. A continuación presentan extractos de algunos de los comentarios realizados por el equipo de Astroselling:

“Me parece importante destacar el valor que le ha presentado la herramienta a Astroselling. A la hora de hacer demostraciones de Astroselling, mostrar Astrolytics siempre llama la atención y acelera las ventas.

A los clientes viejos les brindó un valor agregado que aumentó considerablemente las estadísticas de satisfacción.

Astrolytics se convirtió en uno de los módulos principales de Astroselling y, sin dudas, uno de los que más potencial tiene.

Por otro lado, felicitar a Federico y Augusto ya que lograron entender y llevar a cabo los requerimientos específicos y las necesidades identificadas de una manera muy profesional en todo su accionar. “

“El trabajo del equipo de Astrolytics ha sido excelente desde el comienzo hasta la entrega final. Siempre han sido receptivos a las sugerencias y mejoras propuestas y las han implementado.

No hay ninguna duda que el equipo es muy capaz de llevar adelante un proyecto. Desde el inicio del proyecto Astroselling comunicó necesidades de alto nivel, el equipo las interpretó perfectamente y fueron capaces de bajarlas a requerimientos claros y luego funcionalidades que aportan mucho valor a nuestra herramienta.”

El equipo de desarrollo se encuentra increíblemente conforme con la satisfacción del cliente, valorando la buena relación y disposición que fue mantenida a lo largo del transcurso del proyecto.

5.4. Detalle de las lecciones aprendidas

A lo largo del año de proyecto el equipo de desarrollo fue experimentando obstáculos y superarlos para lograr sus objetivos. A partir de la reflexión de estas experiencias se pudieron aprender varias lecciones reusables en proyectos por venir. A continuación se presentan estas lecciones, desglosadas por topicos.

5.4.1. Comunicación

Con respecto a la comunicación interna del equipo, no surgieron problemas para atacar en ningún momento del proyecto. Esto se debe en gran parte a que el equipo está conformado únicamente por dos integrantes, los cuales a su vez trabajaban dentro de la misma oficina. También se mantuvieron reuniones semanales desde el comienzo del proyecto con el tutor, en las cuales se presentaban avances y evacuaron dudas de ser necesario. Esto fue ayudado luego con la implementación del método SCRUM y Kanban, donde se planteaba un fácil seguimiento del trabajo actual de cada miembro cada dos semanas.

Se considera, sin embargo, que si hubo una mejora en el *modus operandi* de la definición y discusión de los requerimientos al momento de llevarlos del *backlog* al *To-Do* semanal. Ocurrió en las primeras iteraciones de *sprint* que se implementan funcionalidades de manera distinta a la que el otro miembro había entendido que serían implementadas. Este traspié llevó al retraso de un par de funcionalidades al comienzo del desarrollo, pero no decayó en consecuencias graves. Esto llevó a la comprensión de que había ciertas ambigüedades al momento de discutir las *user stories* que se llevaron a cabo, y de a poco el equipo fue entrenándose a sí mismo para cubrir estos detalles y concretarlos de maneras apropiadas y no ambiguas.

5.4.2. Ingeniería de requerimientos

El equipo fue realizando muchas correcciones a la inferencia de requerimientos a lo largo de la ejecución del proyecto. No solo porque los cambios del contexto del desarrollo lo ameritaban, digase, inferir requerimientos sin y con un prototipo usable, sino que también el equipo se percató de que debía cubrir ciertas bases importantes de validación de requerimientos. Con esto en mente, a lo largo del proyecto se fueron

ideando e implementando formas de obtener el *feedback* del usuario de manera crítica y regular, con tal de apuntar el desarrollo en la dirección correcta. Paso de realizarse únicamente *brainstormings* con el cliente, a tener reuniones con los usuarios finales, e incluso ya cerca de la finalización del proyecto una beta cerrada para usuarios específicos con un módulo de *feedback* implementado en la aplicación.

Los integrantes consideran que esta evolución de los métodos de inferencia y validación de requerimientos fue clave, y es algo a tener en cuenta en cualquier tipo de desarrollo ágil que presente requerimientos no muy concretos en un inicio.

5.4.3. Gestión

Con respecto a la gestión del desarrollo, el equipo se encuentra conforme con cómo fue sobrellevada a lo largo del proyecto. Se utilizaron herramientas nuevas como Toggl para realizar un seguimiento de las horas de trabajo, lo cual sirvió para realizar una revisión personal de las horas de cada miembro, junto con la posibilidad de utilizar esta información como métricas que permitan seguir mejorando la gestión. El equipo siente que el objetivo que fue planteado con el ciclo de vida prototipación evolutiva de mejora constante fue logrado con éxito.

Cabe destacar la adaptación que el equipo logró con el proyecto, puntualmente importante con los eventos transcurridos en el año 2020. Un ejemplo de esto es la decisión de reorientar la obtención de *feedback* de los usuarios, cambiando de instancias físicas puntuales (reuniones) a un proceso no presencial a través de la aplicación misma.

Una carencia que el equipo considera haber tenido, es que no realizó un mayor énfasis en la obtención de métricas del proyecto a lo largo del mismo, dado que estas aportarían al *feedback* iterativo. Esta falta de énfasis puede deberse a que era un equipo de solamente dos personas.

5.4.4. Desarrollo

El desarrollo de Astrolytics presentó sus desafíos, los cuales el equipo se enorgullece de considerar que pudieron ser sobrellevados exitosamente. Nunca se había trabajado con volúmenes de datos tan grandes, y se considera que se tomaron en

cuenta los riesgos de *performance* que involucra, y también se realizaron las decisiones apropiadas para evitarlos.

Hubo una funcionalidad que se desearía haber ideado e implementado antes, para poder pulir más en el proyecto, el Monitor de Astrolytics. En futuros proyectos que involucran secuencias de pasos automatizados se tendrá en cuenta la monitoreabilidad del proyecto desde un comienzo.

5.4.5. Arquitectura

El equipo encontró interesante realizar una separación de responsabilidades y trabajar en módulos de software definidos, coordinados entre sí a través de interfaces REST. No surgieron mayores problemas al trabajar con este tipo de arquitectura. Con respecto al *debugging*, no hubo mayores problemas al momento de encontrar la raíz de defectos en el código gracias a que el equipo prestó atención a mantener un nivel de trazabilidad importante en toda la aplicación.

5.4.6. Tecnología

Con respecto a los objetivos personales relevantes a las tecnologías utilizadas, el equipo buscó un balance entre tecnologías familiares y nuevas al momento de elegir cuales utilizar, con el objetivo de poder completar el proyecto de manera satisfactoria, manteniendo un nivel de desafío y mejora personal sustancial.

Como se menciona en secciones previas del documento, uno de los módulos de Astrolytics fue originalmente implementado en NodeJS, un *framework* de desarrollo en el cual ninguno de los dos integrantes poseía experiencia sustancial. El desarrollo del módulo comenzó a estancarse y el equipo decidió cambiar a una tecnología más conocida por ambos como lo es Flask. El equipo considera que esta decisión fue apropiada y la correcta a realizar, ya que no llevo demasiado tiempo traspasar las funcionalidades existentes en ese momento a un proyecto nuevo, para retomar el desarrollo desde ese punto. En algún momento futuro, en algún otro proyecto, se desea ahondar y obtener conocimientos sobre NodeJS, al punto de poder manejarlo a nivel de industria.

Más allá del traspié anterior, el equipo encontró interesante la orquestación de un conglomerado de módulos, con los desafíos que implicó.

5.5. Líneas de trabajo para el futuro

Astrolytics continuará en desarrollo luego de la cesión de derechos, creciendo para conformarse más a las expectativas de Astroselling y sus clientes. Con este objetivo en mente, se planteó un *roadmap* que detalla en manera general los planes a futuro para la aplicación.

5.5.1. Soporte para múltiples regiones

Astroselling desea en un futuro expandir su área de soporte para Astrolytics al resto de los países en los que actualmente opera. Estos están conformados por Argentina, Brasil y Paraguay. Afortunadamente, el equipo de desarrollo estuvo al tanto de este deseo durante la programación de la solución, ajustando detalles para que el pasaje a un modelo multinacional fuera lo más indoloro posible. La fecha estimada de trabajo para esto es a finales de 2020. (Q4 2020)

5.5.2. Sistema Distribuido de Scrapeo

Como consecuencia de la línea de trabajo previamente mencionada, surge la necesidad de incorporar un sistema de distribución de scrapeo y almacenamiento de datos para Astrolytics. Esto se debe a que no solamente se estarían agregando países nuevos al plan de scrapeo diario, sino que estos poseen un volumen de datos mucho más grande que Uruguay. Si no se presta atención a la *performance* para realizar este avance, el proceso de scrapeo podría ralentizarse de manera problemática. Se espera que esta funcionalidad sea elaborada al terminar los ajustes para soportar múltiples países, a principios del 2021 (Q1 2021)

5.5.3. Servicio de Suscripción

Astroselling considera que la funcionalidad que proporciona Astrolytics podría ser efectiva como un producto *standalone*, por lo que sugirió la posibilidad de aplicar un servicio de suscripción para el uso de las funcionalidades del producto, sin necesidad de ser cliente previo de Astroselling. Esto a su vez ayudaría a publicitar su producto principal que posee su mismo nombre, Astroselling. Desean planificar para este asunto a finales de la internacionalización del producto, por lo que se espera sea atacado a principios de 2021. (Q1 2021)

6. Referencias bibliográficas

[1] Platzi, “Qué es Frontend y Backend,” Jun. 2018. [Online]. Available: https://platzi.com/blog/que-es-frontend-y-backend/ Accessed on: Jun. 30, 2020.

[2] MercadoLibre, “Pagina Oficial de MercadoLibre.” Sep. 2020 [Online]. Available: https://www.mercadolibre.com.uy/ Accessed on: Sep. 28, 2019.

[3] Hotjar, “How to create a simple, accurate user persona in 4 steps without leaving your desk,” Nov. 2018. [Online]. Available: https://www.hotjar.com/blog/user-personas/ Accessed on: Jan. 05, 2020.

[4] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Irvine, 2000.

[5] REST API Tutorial, “Learn REST: A RESTful Tutorial.” Jul. 2019 [Online]. Available: https://www.restapitutorial.com/. Accessed on: Jan. 10, 2020.

[6] L. Bass, P. C. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Reading, Massachusetts: Addison-Wesley, 2012.

[7] Paul Clements *et al.*, *Documenting Software Architectures: Views and Beyond*, 2nd ed. Westford, Massachusetts: Pearson Education, 2010.

[8] W. Suryn, *Software Quality Engineering: A Practitioner’s Approach*, Ilustrada. Hoboken, New Jersey: John Wiley & Sons, 2014.

[9] Scrum Guides , *The Scrum Guide ™*. Jan. 2017. [Online]. Available: https://www.scrumguides.org/scrum-guide.html Accesed on: Jan. 05, 2020.

[10] Atlassian, “What is kanban?,” Jan. 2020. [Online]. Available: https://www.atlassian.com/agile/kanban Accessed on: Jan. 02, 2020.

[11] A. S. for Quality, “What is the Plan-Do-Check-Act (PDCA) Cycle?,” Jan. 2020. [Online]. Available: https://asq.org/quality-resources/pdca-cycle Accessed on: Feb. 03, 2020.

[12] M. Cohn, *Agile Estimating and Planning*. Upper Saddle River, New Jersey: Pearson Education, 2005.

[13] Atlassian, “Five agile metrics you won’t hate,” Jan. 2020. [Online]. Available: https://www.atlassian.com/agile/project-management/metrics Accessed on: May 15, 2020.

[14] W. H. Organization, “Statement on the second meeting of the International Health Regulations (2005) Emergency Committee regarding the outbreak of novel coronavirus (2019-nCoV),” Jan. 2020. [Online]. Available: https://www.who.int/news-room/detail/30-01-2020-statement-on-the-second-meeting-of-the-international-health-regulations-(2005)-emergency-committee-regarding-the-outbreak-of-novel-coronavirus-(2019-ncov) Accessed on: Apr. 03, 2020.

[15] Mike Cohn, *User Stories Applied: For Agile Software Development*, Ilustrada. Boston, Massachusetts: Addison-Wesley Professional, 2004.

[16] Mountain Goat Software, “Advantages of the ‘As a user, I want’ user story template,” Apr. 2008. [Online]. Available: https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template Accessed on: May 04, 2020.

[17] G. G. Schulmeyer, *Handbook of Software Quality Assurance*, 4th, Illustrated. Norwood, MA: Artech House, 2008.

[18] Daniel Kummer, “git-flow cheatsheet.” Oct. 2018 [Online]. Available: https://danielkummer.github.io/git-flow-cheatsheet/ Accessed on: Jan. 13, 2020.

[19] The Economist “Parkinson’s Law.” Nov. 1955 [Online]. Available: https://www.economist.com/news/1955/11/19/parkinsons-law Accessed on: Aug. 20, 2020.

7. Anexos

7.1. Anexo 1 - Investigación inicial

Anexo digital.

7.2. Anexo 2 - *Endpoints* de los Web Services

En este anexo se mostrará información sobre los *endpoints* disponibles para interactuar con Analytics.

Endpoint	**Método HTTP**	**Parámetros**	**Descripción**
generate/			Estos *endpoints* se utilizan para procesar y generar la información utilizada en Astrolytics
listings	GET	-	Este *endpoint* descarga el archivo .zip de la carpeta de Drive que consiguió Scraper y procesa la información.
trends	GET	-	Obtiene para cada categoría la información de trends de Mercado Libre
seller-search	GET	-	Genera el índice de búsqueda de vendedores, para no tener que buscar por Mercado Libre para cada consulta del *frontend*.
seller-ranking	GET	-	Genera el *ranking* de vendedores.
favorites-data	GET	-	Obtiene la información del día de stock/precio de las publicaciones agregadas como favoritas
seller_recommendations	GET	-	Crea las recomendaciones para los usuarios de Astrolytics
user-favorites-store-starting-point	GET	-	Averigua la información de las publicaciones actuales de cada tienda agregada como favorita.
user-favorites-store-ending-point	GET	-	Hace lo mismo que el *endpoint* anterior, pero cierra la transacción con la diferencia de cantidad vendida/precio que generó el *starting point*, para obtener el *revenue* diario de cada tienda.
categories/			
update	GET	-	Actualiza el árbol de Categorías de Mercado Libre, se ejecuta una vez a la semana.
data/<meli_id>	GET	meli_id: Identificador de la categoría de Mercado	Devuelve la información de la categoría según su identificador de Mercado Libre (*Ranking*, gráficas)

		Libre	
download-excel	POST	category_id: El id de la categoría. Otros campos.	Genera y permite descargar un archivo de Excel con la información de la categoría mandada.
user/<user_id>/			
favorites	PUT	Información de los favoritos del usuario.	Modifica los favoritos del usuario.
favorites	GET	-	Devuelve las publicaciones y tiendas favoritas del usuario.
favorites/items	GET		Devuelve la información de las publicaciones favoritas del usuario.
favorites/stores	GET		Devuelve la información de las tiendas favoritas del usuario.
favorite-data/<meli_id>	DELETE	meli_id: El identificador de la publicación	Borra una publicación de favoritos.
favorite-data/<meli_id>	POST	meli_id: El identificador de la publicación	Agrega una publicación a favoritos.
sellers/			
<seller_id>/data	GET	seller_id: Identificador del vendedor en Mercado Libre	Devuelve la información del vendedor,que se muestra en la página de vendedores
<store_name>/recommendations	GET	store_name: Nombre de la tienda en Mercado Libre.	Devuelve la información de recomendaciones de la tienda dada.
Otros *endpoints*			
stores/search	GET	*Query*.	Devuelve la información de la búsqueda de las tiendas de Mercado Libre Uruguay según el *query* pasado.
seller-ranking	GET		Devuelve el top 100 de los vendedores de Mercado Libre Uruguay.
feedback	POST	name:	Guarda en la base de datos del

		Nombre de quien escribe el *feedback*. email, text: mensaje.	*backend* un *feedback* subido por un usuario. A la misma vez, manda un mail al equipo de Astrolytics y de Astroselling con la sugerencia.
listings/search	GET	Parámetros de búsqueda, query, offset, y cantidad por página.	Devuelve el resultado de búsqueda paginado con el *query* proporcionado. El *backend* hace de pasamanos entre Mercado Libre y el *frontend*.
trends/<meli_id>	GET	meli_id: Identificador de la categoría en Mercado Libre	Devuelve las *trends* de la categoría consultada.
login/astroselling	POST	Correo electrónico, contraseña.	*Endpoint* usado para iniciar sesión en Astroselling. El *backend* se encarga de comunicarse con Astroselling y devolver la información del usuario y de sus tiendas para el *frontend*.

Tabla 7.2.1. *Endpoints* definidos para componente Analytics

7.3. Anexo 3 - Notas de entrevista con CalzadosUY

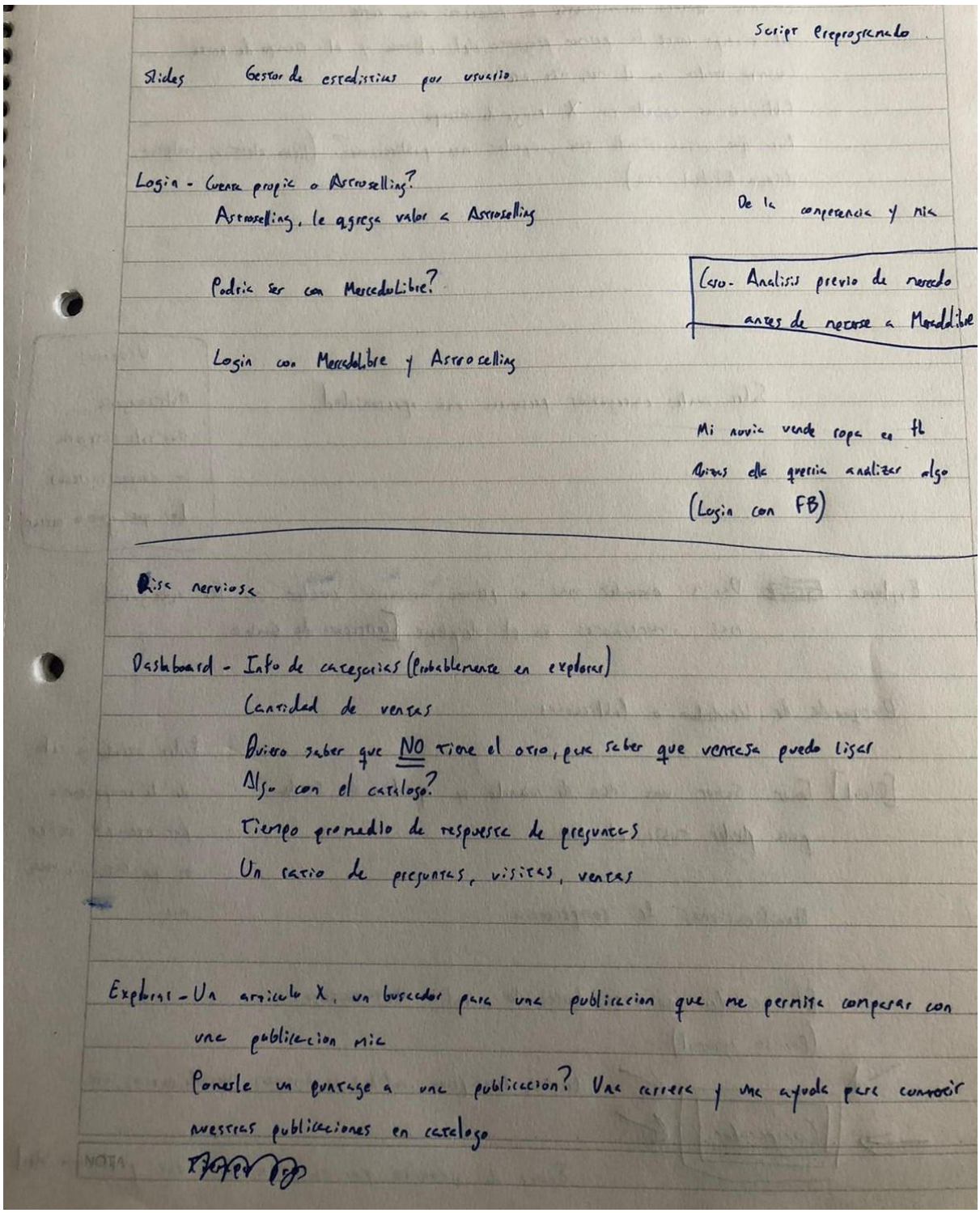


Imagen 7.3.1. Notas de entrevista 1

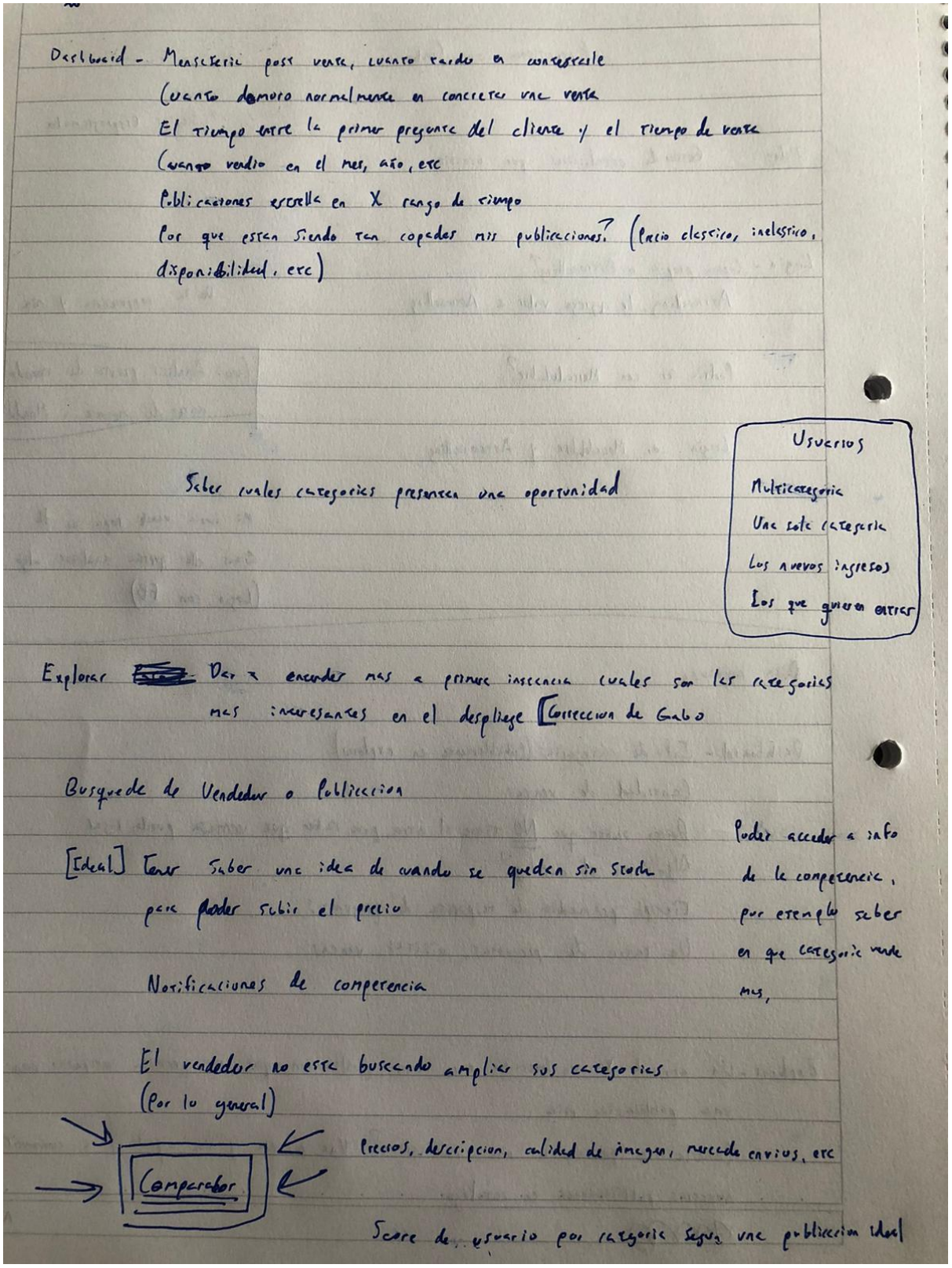


Imagen 7.3.2. Notas de entrevista 2

7.4. Anexo 4 - Manual de usuario

Anexo digital. Este anexo es un manual de usuario entregado al cliente como entregable.

7.5. Anexo 5 - *Feedback* de usuarios

A continuación se presentan varias capturas del *feedback* realizado por parte de los usuarios de Astrolytics, durante el periodo beta. Como recordatorio, este *feedback* se realiza al utilizar el botón de *feedback* en la barra de navegación. Este es luego enviado al equipo de desarrollo vía SendGrid.

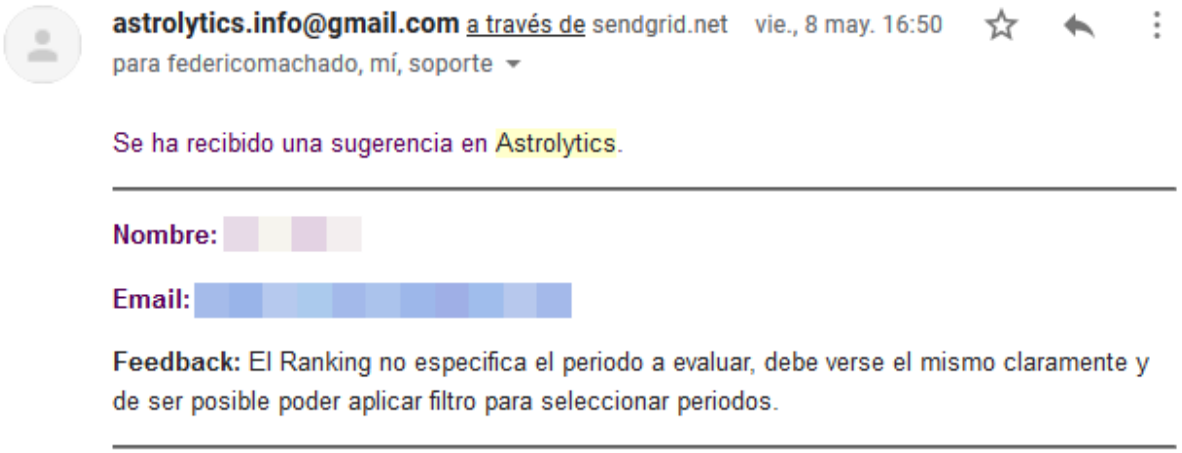


Imagen 7.5.1. *Feedback* de usuario 1

Imagen 7.5.2. *Feedback* de usuario 2

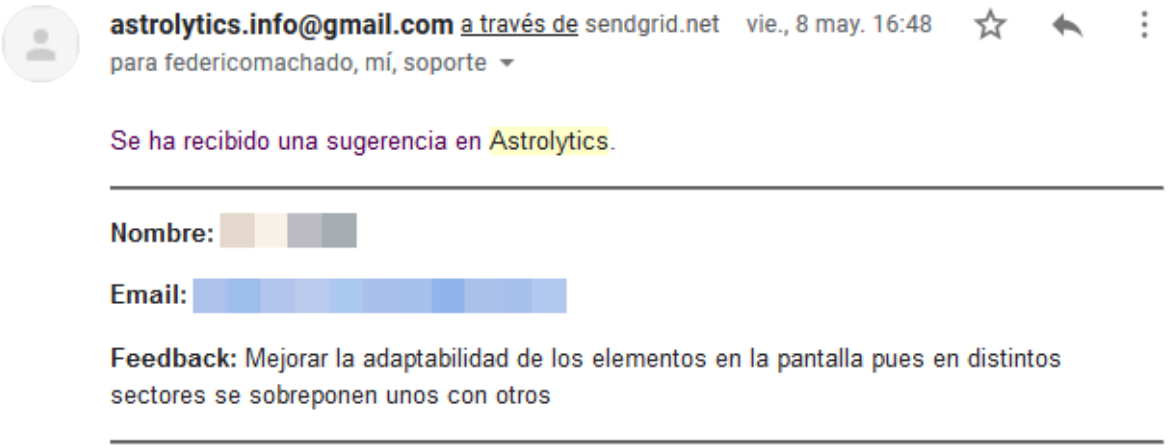


Imagen 7.5.3. *Feedback* de usuario 3

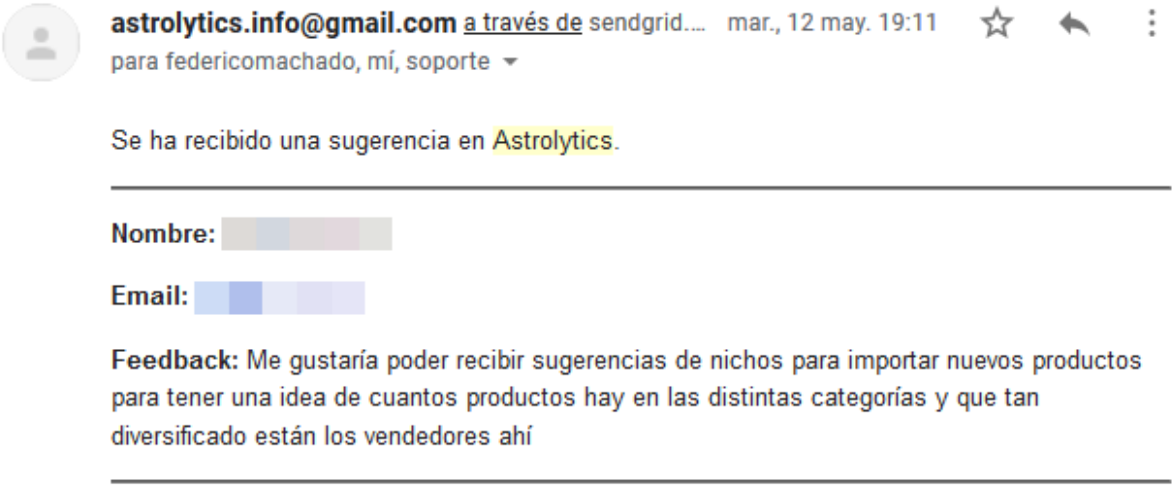
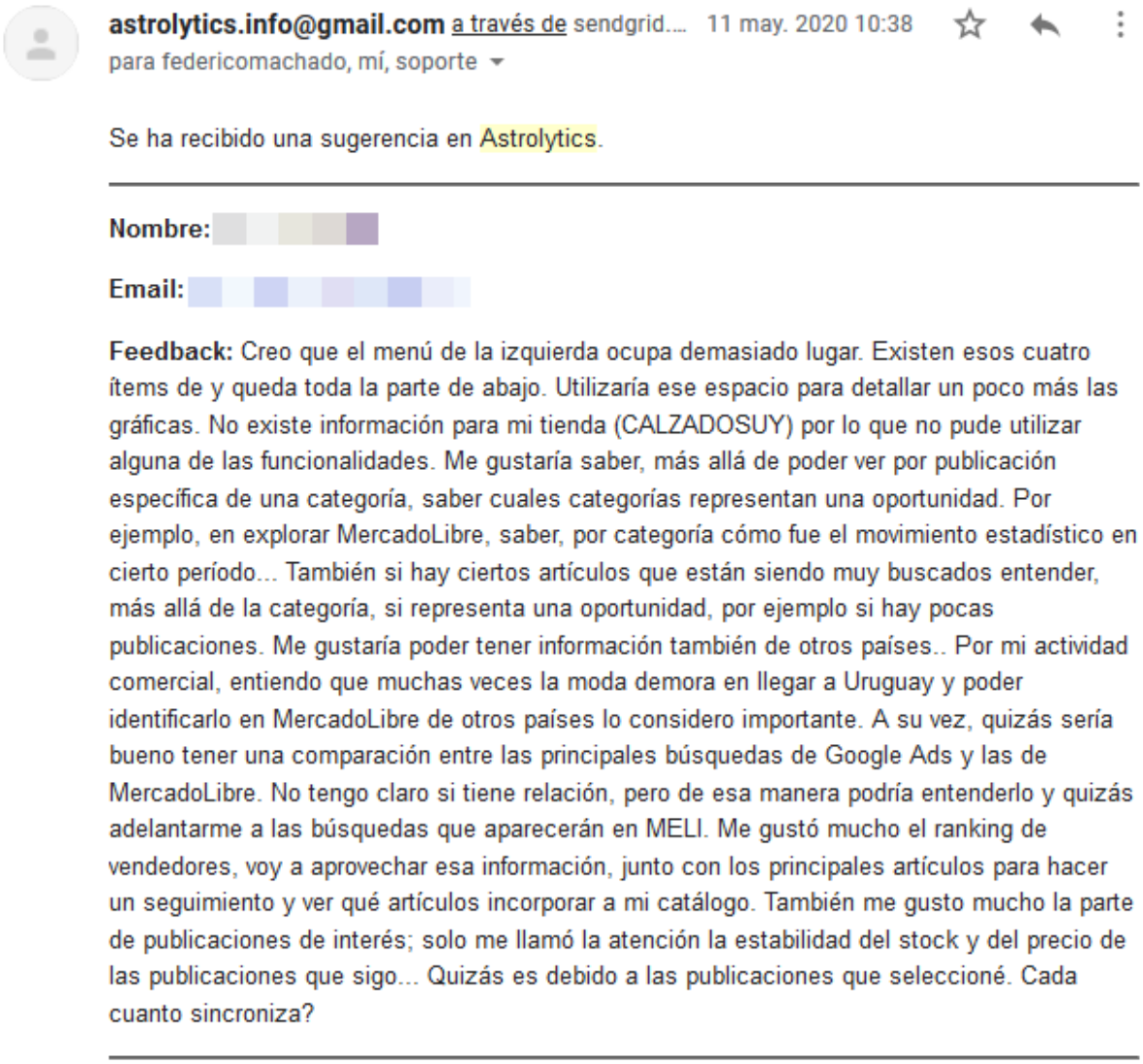


Imagen 7.5.4. *Feedback* de usuario 4



astrolytics.info@gmail.com a través de sendgrid.... 11 may. 2020 10:38

para federicomachado, mí, soporte

Se ha recibido una sugerencia en Astrolytics.

Nombre:

Email:

Feedback: Creo que el menú de la izquierda ocupa demasiado lugar. Existen esos cuatro ítems de y queda toda la parte de abajo. Utilizaría ese espacio para detallar un poco más las gráficas. No existe información para mi tienda (CALZADOSUY) por lo que no pude utilizar alguna de las funcionalidades. Me gustaría saber, más allá de poder ver por publicación específica de una categoría, saber cuales categorías representan una oportunidad. Por ejemplo, en explorar MercadoLibre, saber, por categoría cómo fue el movimiento estadístico en cierto período... También si hay ciertos artículos que están siendo muy buscados entender, más allá de la categoría, si representa una oportunidad, por ejemplo si hay pocas publicaciones. Me gustaría poder tener información también de otros países.. Por mi actividad comercial, entiendo que muchas veces la moda demora en llegar a Uruguay y poder identificarlo en MercadoLibre de otros países lo considero importante. A su vez, quizás sería bueno tener una comparación entre las principales búsquedas de Google Ads y las de MercadoLibre. No tengo claro si tiene relación, pero de esa manera podría entenderlo y quizás adelantarme a las búsquedas que aparecerán en MELI. Me gustó mucho el ranking de vendedores, voy a aprovechar esa información, junto con los principales artículos para hacer un seguimiento y ver qué artículos incorporar a mi catálogo. También me gusto mucho la parte de publicaciones de interés; solo me llamó la atención la estabilidad del stock y del precio de las publicaciones que sigo... Quizás es debido a las publicaciones que seleccioné. Cada cuanto sincroniza?

Imagen 7.5.5. *Feedback* de usuario 5

7.6. Anexo 6 - Bitácora de sprints

Anexo Digital.

7.7 Anexo 7 - Resultado de encuestas de satisfacción al cliente

5 personas del equipo de Astroselling contestaron la encuesta. Cómo es anónima para que no se vean coaccionados a responder de cierta forma, no se sabe a ciencia cierta qué contestaron ni quienes la contestaron, con la excepción en este último punto de Gonzalo Medeiros y Gabriel Fremd de Astroselling que nos confirmaron de primera mano que contestaron la encuesta.

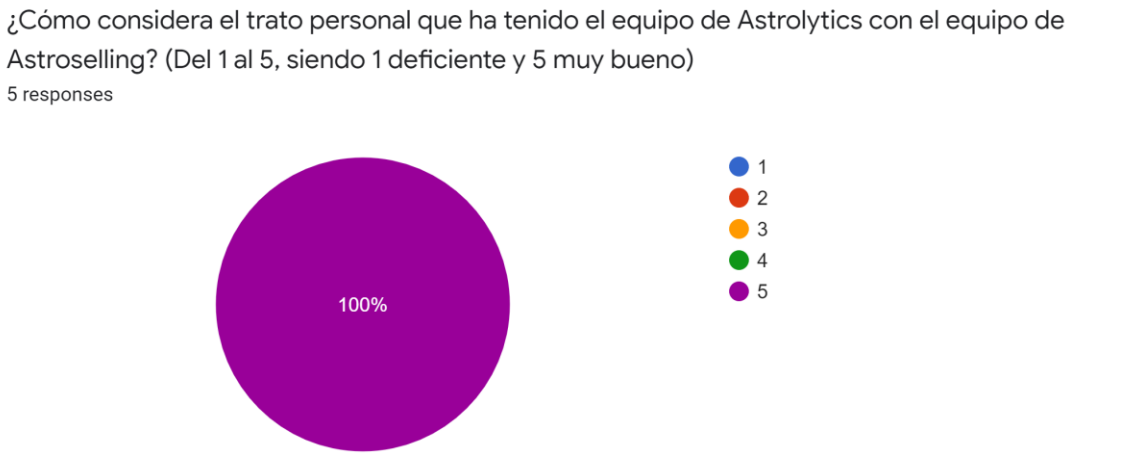


Imagen 7.7.0.1. Respuestas sobre el trato del equipo

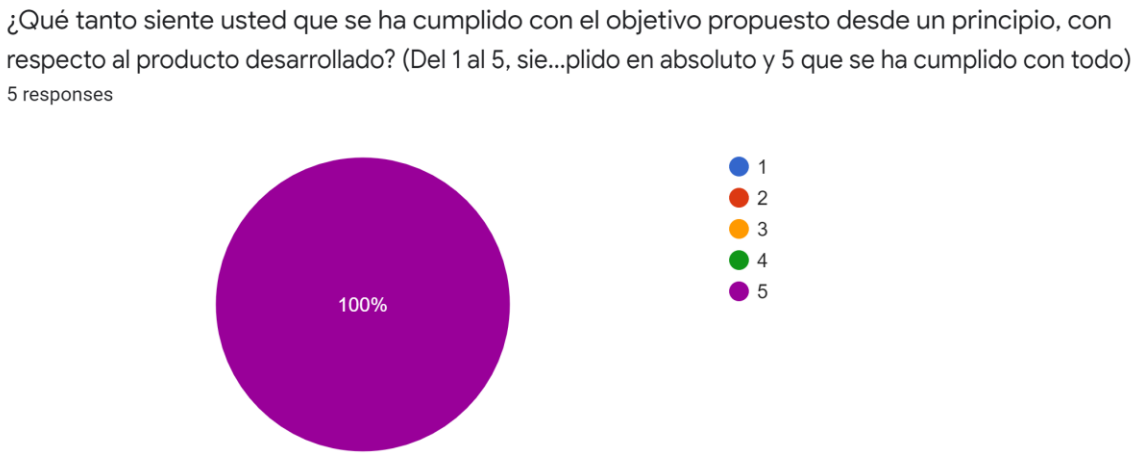


Imagen 7.7.0.2. Respuestas respecto a satisfacción de objetivos del producto

Imagen 7.7.0.3. Respuestas al nivel de comunicación del equipo de desarrollo con el cliente

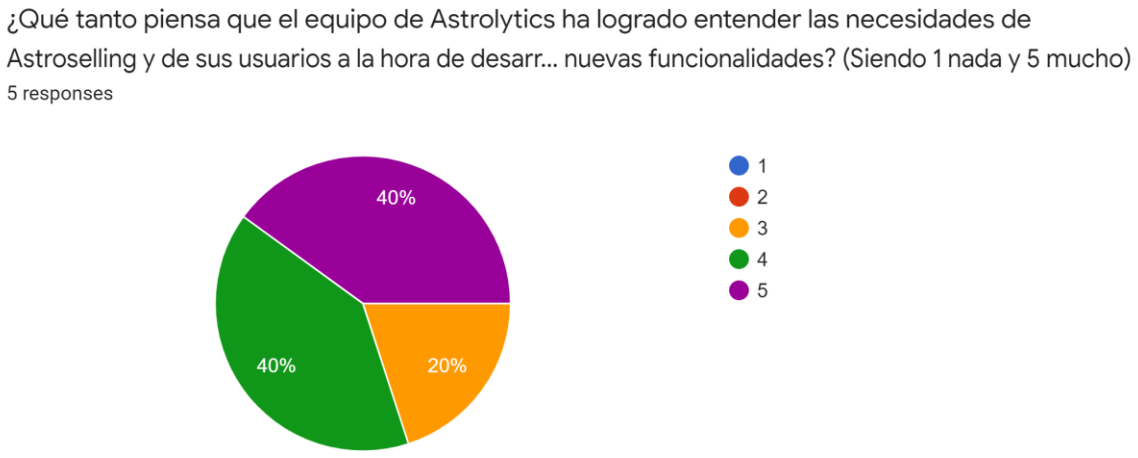


Imagen 7.7.0.4 - Respuestas a la comprensión de necesidades por parte del equipo de desarrollo

A la hora de haber probado la plataforma, ¿Le parece que ha sido estable y sin errores inesperados en su uso? (Del 1 al 5, siendo 1 inutilizable y 5 funcionando correctamente)

5 responses

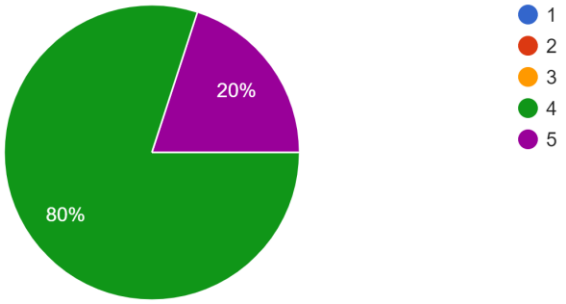


Imagen 7.7.0.5. Respuestas respecto al correcto funcionamiento de la plataforma

¿Considera que la plataforma es navegable de forma rápida? (Del 1 al 5, siendo 1 lenta y 5 rápida)

5 responses

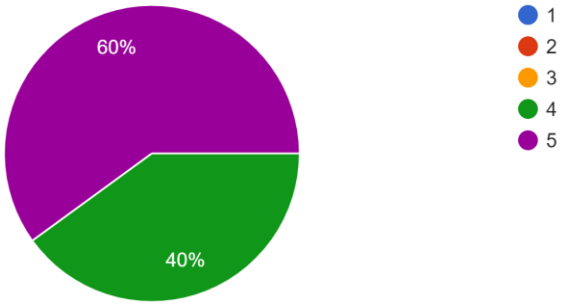


Imagen 7.7.0.6. Respuestas respecto a la agilidad de uso

¿Le parece fácil de usar y de entender en cuanto a las funcionalidades que ofrece y cómo visualizarlas? (Del 1 al 5, siendo 1 para nada intuitivo y 5 muy fácil de usar)

5 responses

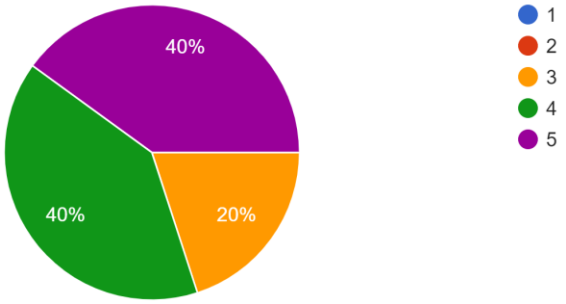


Imagen 7.7.0.7. Respuestas respecto a usabilidad del producto

¿Qué tan útil le parece la herramienta desarrollada? (Del 1 al 5, siendo 1 nada útil y 5 muy útil)

5 responses

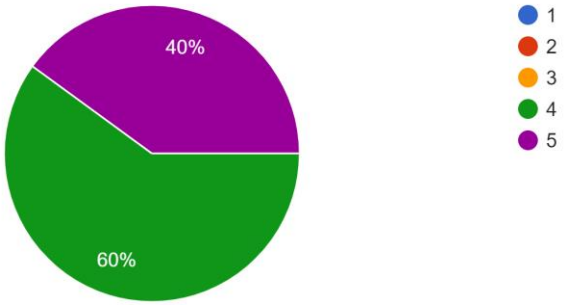


Imagen 7.7.0.8. Respuestas respecto a la utilidad del producto

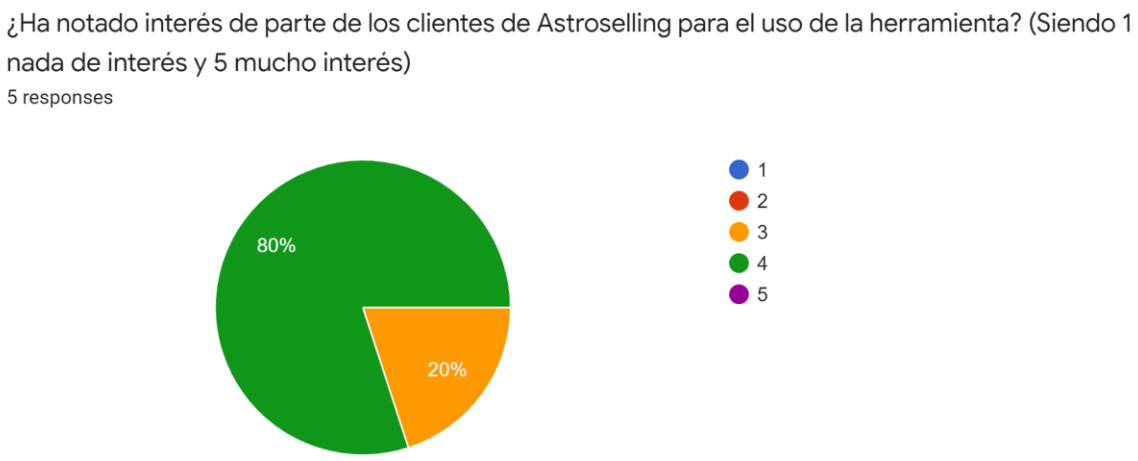


Imagen 7.7.0.9. Respuestas respecto al interés percibido del usuario al producto

Las respuestas de los encuestados en estas preguntas que evalúan del 1 al 5 nos parecen correctas y alineadas con nuestros objetivos. Se destacan entre las respuestas lo bueno de que todos los que han contestado han considerado el trato personal y el cumplimiento lo mejor posible.

En los otros ítems hay un poco más de heterogeneidad en las respuestas. A modo de ejemplo, las preguntas sobre la comunicación sobre las nuevas funcionalidades, el entendimiento de los requerimientos y la usabilidad de la plataforma oscilan entre tres valores distintos de satisfacción, de un puntaje de 3 neutro a un 5 bueno pasando por el 4. Para estos tres casos se notó que solo una persona de las encuestadas contestó con un 3, que tampoco se considera un puntaje malo, ya que se encuentra a la mitad de nuestra escala.

En cuanto a la estabilidad y velocidad de la plataforma, cualidades que siempre han sido consideradas como prioridad, se han obtenido buenas respuestas de parte de los encuestados. Así mismo, hay una percepción de que la herramienta desarrollada es útil, lo cual nos confirma que se tomó una buena decisión al llevar adelante este desafío para nuestro proyecto de grado.

El último ítem a analizar es la percepción de los encuestados en cuanto al interés de los clientes de Astroselling en cuanto a Astrolytics. Se destaca que no hubo ninguna respuesta mala, ni tampoco respuestas muy buenas. El 80% de los encuestados consideró el interés 4 de 5, mientras que uno la consideró un 3 de 5. Si bien este

resultado no es el ideal, se considera que es correcto y satisfactorio para esta etapa del proyecto.

Por otro lado, ¿Considera que Astrolytics ayuda a darle más valor al usuario de Astroselling? fue contestada en un 100% con un sí.

7.7.1. Funcionalidades que les gustaría ver en un futuro

Como esta pregunta era opcional, no la contestaron todos los encuestados. En este apartado se mostrarán las respuestas recibidas de los que sí la contestaron:

"Creo que sería interesante tener disponible la funcionalidad en otros países. Muchos clientes que llegan a la plataforma nos han solicitado utilizar lo que mostramos de la herramienta, pero debemos indicarle que no está disponible. En ese sentido, creo que es interesante tanto para que las personas puedan entender el comportamiento del mercado en su país de residencia, como en otros para poder implementar DropShopping.

Por lo importante que resultó para MercadoLibre, creo que sería interesante poder tener información similar para otros marketplaces, por ejemplo Facebook, así como información de búsqueda de Google Analytics.... Poder cruzar la información de MELI o FB con esa otra búsqueda creo que puede presentar oportunidades (nichos) muy interesantes para los vendedores.

Por otro lado, creo que sería interesante poder presentarle a vendedores que encuentran una categoría-artículo que presenta una oportunidad, DropShippers (importadores que disponibilizan su catálogo para que otros vendan) que tengan esos artículos disponibles para que los vendedores vendan. "

"Sería muy útil alguna herramienta que ayude a los vendedores a gestionar su stock, de forma de no quebrar stock en fechas clave. Por ejemplo, cuando viene un cyberlunes, contando con información de las ventas en los cyberlunes anteriores y el impacto que tuvo, predecir artículos que podría quebrar stock y que por lo tanto no debería promocionar. O, por el contrario, sugerir artículos con mucho inventario que podría ofrecer un descuento atractivo.

Lo mismo podría aplicarse a épocas del año, previo al verano crecen mucho las ventas de artículos de fitness y actividades al aire libre. Considerando esa variable, y proyectando las ventas en los últimos meses, se le podrían hacer sugerencias a los vendedores."

"Ingreso a la plataforma con facebook"

Se cree que estas sugerencias son valiosas y serán contempladas para desarrollos posteriores al proyecto.

7.7.2 Comentarios adicionales acerca de Astrolytics

Al igual que en el apartado anterior, no todos contestaron esta pregunta, solo lo hicieron 2 de los 5 encuestados:

“Me parece importante destacar el valor que le ha presentado la herramienta a Astroselling. A la hora de hacer demostraciones de Astroselling, mostrar Astrolytics siempre llama la atención y acelera las ventas.

A los clientes viejos les brindó un valor agregado que aumentó considerablemente las estadísticas de satisfacción.

Astrolytics se convirtió en uno de los módulos principales de Astroselling y, sin dudas, uno de los que más potencial tiene.

Por otro lado, felicitar a Federico y Augusto ya que lograron entender y llevar a cabo los requerimientos específicos y las necesidades identificadas de una manera muy profesional en todo su accionar. “

“El trabajo del equipo de Astrolytics ha sido excelente desde el comienzo hasta la entrega final. Siempre han sido receptivos a las sugerencias y mejoras propuestas y las han implementado.

No hay ninguna duda que el equipo es muy capaz de llevar adelante un proyecto. Desde el inicio del proyecto Astroselling comunicó necesidades de alto nivel, el equipo las interpretó perfectamente y fueron capaces de bajarlas a requerimientos claros y luego funcionalidades que aportan mucho valor a nuestra herramienta.”

Se agradece a todo el equipo de Astroselling por sus comentarios y por el apoyo brindado a lo largo del proyecto.