Universidad ORT Uruguay

Facultad de Ingeniería

e-Profiler: Generación de campañas publicitarias en base a perfil de usuarios de Mercado Libre

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

Rodrigo Castro - 181983

Pablo Larralde - 162960

Tutor: Álvaro Ortas

2021

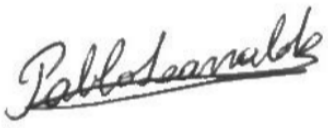
Declaración de autoría

Nosotros, Rodrigo Castro y Pablo Larralde, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizamos el proyecto final de la carrera 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 qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Rodrigo Castro

30/09/2021



Pablo Larralde

30/09/2021

Agradecimientos

En primer lugar queremos agradecer a nuestras familias, por el apoyo emocional y logístico que nos han aportado durante el tiempo en el que se realizó el proyecto.

Seguidamente queremos agradecer a la empresa Mercado Libre, representada en esta ocasión por Rafael Hermida, quien nos ha brindado la oportunidad de realizar este proyecto y lograr combinar nuestro trabajo con el ambiente académico.

No queremos dejar de agradecer a los referentes técnicos Luis Argerich y Gianmarco Cafferata, cuyos aportes han sido de gran valor para este proyecto.

A nuestro tutor, Álvaro Ortas, quien nos ha aportado su experiencia para guiar al equipo en el transcurso del proyecto y nos ha permitido llegar a elaborar un producto alineado a los requerimientos académicos.

Abstract

En la actualidad el e-commerce se enfrenta a un entorno cada vez más competitivo y que afronta distintos desafíos, dentro de los cuales se pueden destacar la adquisición y retención de clientes. El hecho de ofrecer una experiencia más personalizada para dichos clientes hará que se sientan más contemplados y en definitiva, terminen concretando una compra. Por otro lado, actualmente disponemos de herramientas más avanzadas de inteligencia artificial, las cuales pueden ser aprovechadas para contribuir a generar esa experiencia más personalizada.

Nuestro proyecto va en el sentido con lo expresado anteriormente, y tuvo el objetivo de evolucionar un sistema actual de recomendaciones del e-commerce Mercado Libre y afrontar el desafío de clasificar sus usuarios en perfiles. Creímos que si entendemos las similitudes entre ellos (en base a su comportamiento), podremos llegar a ofrecer otros productos relacionados a compras realizadas por otros usuarios del perfil, tomando un criterio más holístico de recomendación que el utilizado actualmente (que recomienda productos estrechamente relacionados a los previamente adquiridos por cada usuario en particular, sin considerar a otros).

La solución que entregamos incluye la identificación y agrupamiento de usuarios en perfiles, envío de recomendaciones automatizadas para los usuarios de cada perfil y una interfaz gráfica para gestionar y monitorear el envío de recomendaciones.

Debido a la incertidumbre manejada inicialmente en el proyecto, el tiempo lo dividimos en dos etapas, en la primera (Discovery), nos concentramos en entender el problema, relevamos los requerimientos, evaluamos tecnologías, implementamos pruebas de concepto y prototipos para presentarle al cliente, entre otras tareas. Una vez cumplidas las tareas de la etapa de Discovery comenzamos con la etapa de Delivery, en la cual desarrollamos la solución y la entregamos al cliente.

Para la realización del proyecto utilizamos varias metodologías de trabajo (entre ellas Scrum-Ban (Scrum[1] + Kanban[2]), BDD[3] y Crisp-DM[4]) que fueron personalizadas acorde al contexto y posibilidades del equipo. La combinación y personalización de las metodologías nos permitió acompañar las diferentes problemáticas de cada etapa pudiendo así colaborar con el objetivo final.

Palabras clave

BDD; Crisp DM; Data Mining; Desarrollo Ágil; E-Commerce; Fury; Fury Data App; Java; Kanban; Mercado Libre; Perfiles de Usuario; Productos Recomendados; React; Scrum; Scrumban.

Glosario

Aplicación: En informática, el software de aplicación es un tipo de software de computadora diseñado para realizar un grupo de funciones, tareas o actividades coordinadas para el beneficio del usuario.

Algoritmo: Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.

Artefacto: En ocasiones un artefacto puede referirse a un producto terminado, como el código o el ejecutable.

Backend: En el diseño de software el backend end es la parte del software que procesa las entradas desde el frontend.

Base de datos: Programa capaz de almacenar gran cantidad de datos, relacionados y estructurados, que pueden ser consultados rápidamente de acuerdo con las características selectivas que se deseen.

Beta continuo: Filosofía de trabajo para representar una modalidad en donde constantemente se está trabajando en busca del cambio junto con una la mejora continua en el desarrollo de las aplicaciones.

Brainstorming: La lluvia de ideas, también denominada tormenta de ideas, o "brainstorming", es una herramienta de trabajo grupal que facilita el surgimiento de nuevas ideas sobre un tema o problema determinado.

Branch: Rama o bifurcación en las versiones de código dentro de un sistema de control de versión.

Bug Fixing: Tarea que comprende la corrección de bugs encontrados en el sistema.

Branching Model: Tipo o estrategia para generar las distintas branches o ramas dentro de un control de versión.

Branch Release: Rama utilizada dentro de un sistema de control de versiones para la generación de una versión de Release o entrega.

Categoría: Denominación al conjunto que contiene uno o más productos agrupados en base a reglas de negocio.

Clasificación IEEE: El Instituto de Ingenieros en Electricidad y Electrónica (IEEE, que se pronuncia “I, triple E”) es un organismo profesional para aquellos que trabajan en los campos de la electrónica y de la ingeniería eléctrica y se dedican a promover la innovación tecnológica y crear estándares.

Collaborative filtering: Es una técnica usada para construir recomendaciones personalizadas en base a ítems de interés.

Coverage: En aseguramiento de la calidad, porcentaje de cobertura de líneas de código cubiertas por tests automatizados.

Crisp-DM: Metodología para proyectos de data mining.

Delivery: Etapa de entrega de los incrementos de software o de valor para el cliente.

Discovery: Etapa del desarrollo de software que permitió al equipo relevar requerimientos, determinar la tecnología a utilizar, etc.

Frontend: En diseño de software el frontend es la parte del software que interactúa con los usuarios.

FDA (Fury Data App): Tipo de aplicación para el desarrollo de proyectos de Data Mining dentro de la suite de Fury.

Fury: Suite de aplicaciones provistas por el equipo de desarrollo interno de Mercado Libre abocado a la construcción, mantenimiento y despliegue para diversas aplicaciones para su ecosistema.

Git Flow: Sistema que se destaca en poder ayudar en la organización de la versión de un código.

Google Drive: servicio de alojamiento de archivos que fue introducido por la empresa estadounidense Google.

Head Branch: Rama principal dentro de un sistema de control de versión.

Hermes: Sistema perteneciente al equipo de Notifications de Mercado Libre que permite envío de comunicaciones a usuarios de la plataforma.

Interfaz gráfica: En informática llamamos Interfaz Gráfica de Usuario al software que permite la interacción con la máquina de manera gráfica, esto es con elementos como botones, ventanas, iconos, enlaces, etc.

Ítem (o producto): Denominación a los artículos que se comercializan dentro de la plataforma de e-commerce Mercado Libre.

Jacoco: Es una herramienta para analizar la cobertura de pruebas unitarias. Después de usarla para ejecutar pruebas unitarias, puede indicar qué partes del código se prueban mediante pruebas unitarias y qué partes están sin probar.

Jupyter notebook: Es una interfaz web de código abierto que permite la inclusión de texto, vídeo, audio, imágenes así como la ejecución de código a través del navegador en múltiples lenguajes.

JUnit: Es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Línea base: Una línea base o baseline según el estándar de la IEEE, es una especificación o producto que ha sido revisado formalmente, sobre el que se ha llegado a un acuerdo, y que de ahí en adelante servirá como base para un desarrollo posterior que puede cambiarse solamente a través de procedimientos formales de control de cambios.

Linters check: Consiste en verificar el estilo de codificación en relación a la sintaxis del código.

Machine Learning: Es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello. Una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos para hacer predicciones.

Matriz dispersa: Es una matriz en la que la mayoría de los elementos son ceros. No existe una definición estricta de cuántos elementos deben ser cero para que se considere una matriz.

Mercado Libre: Refiere a la plataforma de e-commerce fundada por Marcos Galperín.

Merge: Dentro de Git Flow, se refiere a la acción de fusionar dos branches diferentes en una única.

Metodología: Conjunto de métodos que se siguen en una investigación científica, un estudio o una exposición doctrinal.

Nasdaq: Es el segundo mercado de valores y bolsa de valores automatizada y electrónica más grande de los Estados Unidos, siendo la primera la Bolsa de Nueva York, con más de 3800 compañías y corporaciones.

Pull Request: Acción requerida dentro de Git Flow para incorporar cambios a otra rama de otro repositorio.

Release: Publicación final de una determinada versión para una aplicación de software.

Release candidate: Una versión candidata a definitiva, candidata a versión final o candidata para el lanzamiento, comprende un producto preparado para publicarse como versión definitiva a menos que aparezcan errores que lo impidan.

Repositorio: Es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software.

Scope: Espacio de memoria e infraestructura a demanda que permite la ejecución y control de una determinada versión de software en un determinado lugar que permite ser utilizado en tiempo de ejecución.

Stakeholders: Personas o grupos de personas interesados en un fin en común.

Technical Leader: Es un desarrollador de software, responsable de liderar un equipo de desarrollo y responsable de la calidad de sus productos técnicos. Un Tech Lead establece una visión técnica con el equipo de desarrollo y trabaja con ellos para conseguir el objetivo.

OpsGenie: Sistema de alertas que agrupa automáticamente alertas relacionadas de diversos sistemas en un único incidente según las condiciones especificadas.

Producto: Resultado obtenido mediante la aplicación de diversas técnicas y métodos para las etapas de construcción y diseño del software.

Workflow: El flujo de trabajo es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

Índice

1. Introducción

1.1. Conformación del equipo

El equipo del proyecto e-Profiler estuvo integrado por Rodrigo Castro y Pablo Larralde, ambos cursando la carrera de Ingeniería en Sistemas y además empleados de la empresa Mercado Libre desde hace varios años.

Somos compañeros de trabajo, charlas de pasillo y hemos conformado grupos para la realización de trabajos obligatorios a lo largo de la carrera. Por lo anterior, y la confianza mutua que existe entre nosotros, optamos por unir fuerzas y desarrollar en conjunto el proyecto final de grado en la Universidad ORT Uruguay.

La contraparte de negocio de la empresa para este proyecto fue conformada por el Ing. Rafael Hermida (en adelante “cliente”), actual Country Manager de Mercado Libre en Uruguay. Rafael cumplió el rol de Product Owner, aportando contexto sobre el negocio, validando requerimientos y dando feedback sobre el producto entregado.

En cuanto a la contraparte técnica del equipo dentro de la empresa, principalmente fue integrada por Luis Argerich, Máster en Data Science y Data Scientist de Mercado Libre. Con la experiencia y conocimiento que nos aportó, fuimos capaces de sortear varios problemas y desafíos que fueron surgiendo.

Por último, pero no menos importante, Álvaro Ortas, nuestro tutor, nos brindó toda su experiencia y conocimiento para poder encarar el proyecto desde el punto de vista académico y nos orientó durante todo el proceso.

1.2. Descripción del cliente

Nuestro proyecto fue desarrollado para la empresa Mercado Libre, una compañía de e-commerce fundada por Marcos Galperín en el año 2000. Mercado Libre es conocida por ser una de las primeras compañías del mercado fintech que cotizan sus acciones en el Nasdaq.

La empresa opera actualmente en latinoamérica y desde sus comienzos se ha abocado a construir y evolucionar un ecosistema que permita transformar el e-commerce en una experiencia completa para el usuario.

En la siguiente imagen mostramos de forma resumida las cinco líneas de negocio que integran el ecosistema de Mercado Libre, además de una breve descripción de cada una de ellas.

Figura 1.1: Líneas de negocio de la empresa Mercado Libre

Mercado Libre: Marketplace principal de la empresa donde los usuarios compran, venden artículos, además de clasificados. Nuestro proyecto se centrará sobre esta parte del ecosistema.

Mercado Pago: Plataforma de pagos de la compañía, en la cual los usuarios pueden realizar diversos tipos de pagos, además de transferencias de dinero, solicitud de créditos, entre otras.

Mercado Envíos: Plataforma de envíos que permiten definir una red logística que habilita que los productos lleguen desde los vendedores o depósitos a los clientes.

Mercado Ads: Plataforma de publicidad que permite publicar anuncios de productos en internet.

Mercado Shops: Plataforma de tiendas que permite realizar integraciones de sitios que tercerizan sus servicios a Mercado Libre, de forma de poder utilizar los beneficios de la plataforma de e-commerce.

Como ya mencionamos en el capítulo anterior, nuestro contacto para las cuestiones de negocio fue el Ing. Rafael Hermida, Country Manager de Mercado Libre para Uruguay.

1.3. Objetivos del proyecto

En esta sección enumeramos los objetivos del proyecto establecidos en conjunto con el cliente y el tutor del proyecto.

Objetivo 1: Finalizar y aprobar el proyecto de Ingeniería en septiembre de 2021 con una calificación satisfactoria para el equipo.

Este objetivo lo mediremos con el valor de la calificación brindada por los correctores una vez finalizada la presentación final.

Objetivo 2: Elaborar un producto que cumpla con los requerimientos acordados con el cliente y entregárselo en el mes de septiembre de 2021.

La medición de este objetivo se dará en un mediano o largo plazo (aproximadamente seis meses) una vez que el sistema esté disponible en producción. Utilizaremos la métrica de frecuencia de compra acordada con el cliente. Para ampliar información sobre las métricas dirigirse al Anexo 6: Definición de métricas.

Objetivo 3: Aplicar conocimientos de gestión y tecnologías aprendidos durante la carrera en beneficio de alcanzar los objetivos de producto.

Este punto lo mediremos considerando la retrospectiva interna del equipo, además de la apreciación del Tutor del proyecto una vez finalizado el proyecto.

Objetivo 4: Ganar experiencia y conocimiento interactuando un año con un cliente real, pudiendo obtener aprendizajes sobre sus exigencias, modo de trabajo y conocimiento del negocio.

Este punto lo mediremos considerando la retrospectiva interna del equipo, además de la apreciación del Tutor del proyecto una vez finalizado el proyecto.

Objetivo 5: Dar visibilidad del producto elaborado al resto de Mercado Libre, dando a conocer las herramientas y convirtiéndonos en referentes.

Para determinar si fue cumplido o no este objetivo, además de la retrospectiva interna del equipo, consideramos las apreciaciones y/o críticas recibidas sobre el producto y el valor que aporta para la compañía.

En el capítulo de las conclusiones de este documento, volveremos a mencionar los objetivos pero con el enfoque de determinar en qué medida fueron cumplidos y de esta forma poder concluir si el proyecto fue exitoso.

1.4. Estructura del documento

Con el objetivo de guiar al lector a lo largo del documento y facilitar su comprensión, presentamos los capítulos que componen este documento y una breve descripción de cada uno de ellos.

Como en el documento manejamos nomenclatura técnica y específica del negocio, incluimos un glosario para especificar su significado en el contexto de este proyecto.

Contexto y Problema

En este capítulo realizaremos una explicación del origen del proyecto, como también sobre el contexto y problema dentro del ecosistema de Mercado Libre, además de especificar cómo nos condicionó a la hora de pensar, diseñar e implementar la solución final.

Solución

Aquí detallaremos la solución propuesta por el equipo para el problema a resolver en el contexto planteado previamente. Mencionaremos cómo la misma le aportará valor agregado al cliente, haciendo una comparación entre la solución actual con respecto a la nuestra.

Dentro del detalle de la solución, incluiremos el marco teórico sobre el cual se apoyan las técnicas utilizadas en la misma y cómo las mismas pueden aportar un diferencial en comparación con las actuales.

En esta sección detallaremos los requerimientos funcionales, no funcionales, la arquitectura, y cómo ésta satisface a los últimos, además de explicar cuáles fueron las tecnologías utilizadas en el desarrollo.

Metodología

El uso y la justificación de una o varias metodologías aplicadas en un proyecto es clave para el éxito del mismo, debido a esto, en este capítulo explicamos qué metodologías utilizamos en cada fase del proyecto. Detallaremos cuál o cuáles fueron las metodologías utilizadas, por qué y cómo las adaptamos para nuestro equipo y naturaleza del proyecto.

Ejecución del proyecto

En este apartado incluimos detalles del proceso de desarrollo de la solución, reuniones y temas referidos al seguimiento del plan de proyecto. También se incluyen fundamentaciones de las decisiones tomadas durante el desarrollo del mismo en relación a las desviaciones respecto del plan original.

Lecciones aprendidas y Conclusiones

Desde etapas iniciales hasta las finales, este proyecto dejó al equipo muchas enseñanzas y lecciones aprendidas sobre diversos temas. Por eso es que en este capítulo resumimos las que nos parecen más interesantes y deseamos poder compartirlas con otros equipos que deseen llevar adelante un proyecto de similares características.

Next Steps

Debido a que se trataba de un proyecto académico de duración limitada a un año, se establecieron determinadas funcionalidades para el sistema que se incluirán en un futuro cercano (posterior a la instancia académica).

Nos manifestaron desde la empresa Mercado Libre que existe interés del equipo de negocio de poder seguir evolucionando la herramienta, por lo que dejamos establecidos los próximos pasos a seguir para continuar con el proyecto.

Referencias bibliográficas

En esta sección se detallan las referencias bibliográficas que han sido utilizadas en el proyecto, con el formato establecido por la Universidad ORT Uruguay.

Anexos

En la sección de Anexos, incluimos documentos que amplían la documentación del proyecto. Éstos son referenciados desde el documento principal y permiten al lector acceder a más información o documentos del proyecto.

2. Contexto y problema

2.1. Descripción del contexto

Como ya comentamos anteriormente, Mercado Libre es una empresa que opera en su totalidad en América Latina, y dentro de su ecosistema se registran aproximadamente 15 millones de transacciones diarias, dispone de más de 182 millones de usuarios registrados que generan este gran número de operaciones diariamente.

Dentro de todo este ecosistema, como ya comentamos también anteriormente la plataforma ofrece diversas funcionalidades al usuario, pudiendo destacar el hecho de comprar productos, coordinar el envío de los mismos, comparación de precios, pagos, etc.

Estos usuarios que realizan compras dentro de la plataforma y sabemos que se aplican estrategias de cross-selling, en las cuales la empresa ofrece productos similares a los que ya adquirió con el objetivo de ofrecerle una mejor experiencia y lograr aumentar el valor del ticket de compra.

2.2. Problema que se quería resolver

En una plataforma de e-commerce que desea estar a la vanguardia de nuevas formas de acercar la propuesta de valor y la mejor experiencia al cliente es de vital importancia poder contar con herramientas y técnicas que permitan soportar las necesidades de negocio.

El problema concreto que se quería resolver radica en que nuestra contraparte de negocio Rafael Hermida tiene la intención de que Mercado Libre genere una mejor experiencia al usuario, desde el punto de vista de ofrecerle productos más adaptados a su perfil y no solamente ofrecer productos que estén estrechamente relacionados. Esto es, no desea continuar ofreciendo sólo productos relacionados a una compra anterior de un usuario puntual, sino que desea entender al usuario como un participante de un perfil o grupo que comparte intereses con otros y por ende puede estar interesado en productos similares a los del grupo, aunque esto suponga recomendarle productos no estrechamente relacionados con compras anteriores.

Rafael nos comentó que en el pasado con su equipo se intentó hacer este análisis en forma manual, pero dado el contexto que mencionamos en el punto anterior (escala del problema) fue imposible llegar a ningún resultado útil.

Para nosotros uno de los principales retos fue tratar de entender cómo podíamos generar recomendaciones basadas en productos que pudieran tener relación directa o indirecta sobre productos adquiridos en la plataforma y que sobre estos se pudieran recomendar nuevos productos que puedan ser de interés para el usuario (incluso sorprenderlo con una recomendación de un producto que no creía que podía interesarle) y así poder concretar nuevas ventas en la plataforma.

Rafael nos presentó la problemática de recomendar productos previamente visitados como algo obsoleto y con baja aceptación por parte de los clientes, ya que una vez adquirido un producto, no resulta de interés recomendar lo mismo. A continuación mostramos una captura de pantalla para un ejemplo de cable de cargadores de iPhone en la plataforma.

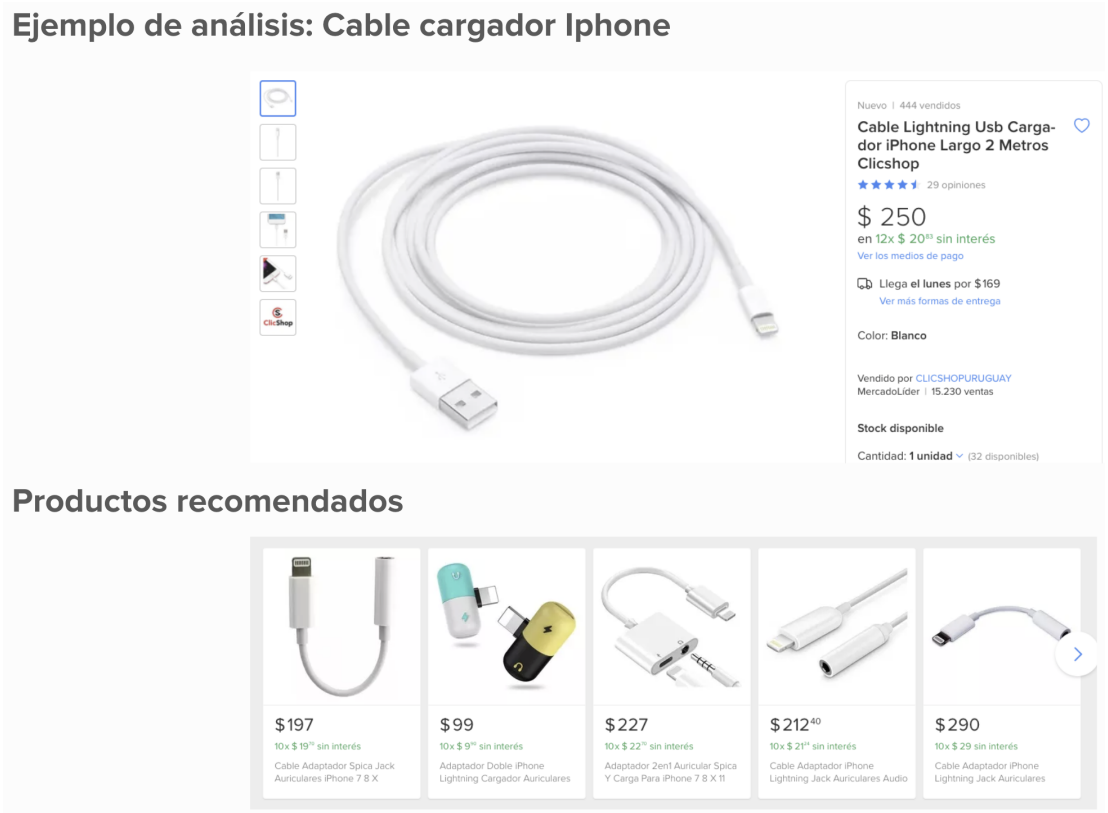


Figura 2.1: Solución actual de recomendaciones de Mercado Libre.

2.3. Alcance del proyecto

Una vez presentado el contexto y problema, describimos el alcance del proyecto, el cual creemos que es de vital importancia para alinear expectativas, considerando las capacidades del equipo y con el objetivo de lograr un resultado exitoso.

Teniendo en cuenta que se trata de un proyecto universitario de un año de duración desarrollado y gestionado únicamente por un equipo de dos integrantes que no disponían de la experiencia en un proyecto de este tipo, se establecieron un conjunto de acuerdos que ayudaron a delinear un límite.

Como ya dijimos, Mercado Libre es una plataforma que opera en varios países de América Latina y por consiguiente existe un gran volumen de información para cada uno de sitios, por esta razón era difícil de manejar el hecho de trabajar con todo el universo de usuarios de Mercado Libre.

En conjunto con nuestro cliente, acordamos comenzar el análisis y el desarrollo del proyecto para un conjunto de usuarios de Mercado Libre Brasil que hayan realizado más de 10 compras durante el año 2020 en categorías distintas. Las restricciones de alcance fueron propuestas por Rafael y aceptadas por el equipo.

Las fundamentaciones provistas por nuestro cliente para cada una de estas restricciones son las siguientes:

- **Usuarios de Mercado Libre Brasil:** este país fue el que registró un porcentaje muy importante de las ventas del año 2020 (ver anexo con gráfico que especifica que brasil registró el mayor volumen de transacciones de la compañía), además de disponer de un set de datos más diverso y menos propenso a outliers.

- **Usuarios que hayan realizado más de 10 compras en categorías de productos distintas:** se fundamenta en descartar usuarios que hayan realizado pocas compras y enfocarnos en usuarios que tengan un nivel de engagement importante (compradores frecuentes) con la plataforma.

- **Usuarios que hayan comprado durante 2020:** consideramos el año anterior al de puesta en producción del proyecto.

3. Solución

En el presente capítulo describimos la solución provista por nuestro equipo al problema planteado, dado el contexto de la compañía de nuestro cliente.

3.1. Marco teórico

Como futuros Ingenieros creemos que debemos estar actualizados, en conocimiento de las soluciones existentes y como no pretendimos reinventar la rueda, nuestra investigación comenzó analizando un problema similar aplicado en otras industrias que son líderes en el mercado actual. En primer lugar comenzamos analizando el recomendador de películas desarrollado por Netflix[5], teniendo en cuenta su contexto y qué herramientas utilizaron.

3.2. Técnicas

A continuación describimos algunas técnicas que nos ayudaron a resolver el problema planteado por nuestro cliente y de qué manera lo logran.

3.2.1. Collaborative filtering

Collaborative filtering es un proceso predictivo utilizado por los motores de recomendaciones, que analizan información sobre elementos similares a otros en base a ciertas similitudes que tiene el primero sobre los demás.

Este proceso permite hacer predicciones automáticas (filtering) sobre los intereses de los usuarios mediante la recolección de sus preferencias o características, de ahí el concepto de colaborativo (collaborative).

La base de este proceso es el enfoque de que si una persona A tiene los mismos intereses que la persona B sobre un determinado tema, es más probable que A tenga los mismos gustos que B.

Dado un ejemplo en donde hay una aplicación directa de este concepto, podría ser en los sistemas de recomendación de programas de televisión en donde se generan predicciones sobre posibles gustos que un usuario podría tener en base a una lista de gustos.

Es importante destacar que las predicciones son específicas para cada usuario, pero usa información provista por varios otros. Este es más específico y difiere del enfoque tradicional de proveer a un determinado usuario una recomendación en base a la cantidad de votos genéricos (no específicos) para cada ítem de su interés.

En el Anexo 8: Alternativas de collaborative filtering, mostramos otra variantes de esta técnica.

3.2.2. Clustering

El análisis de clustering es la tarea de agrupar elementos similares bajo un determinado criterio y clasificarlos en el mismo grupo (o cluster).

Esta tarea no es en sí un algoritmo específico, sin embargo puede ser logrado mediante varios algoritmos que difieren significativamente en el entendimiento sobre cuál es la definición de cluster y cómo encontrarlos de forma eficiente. La noción popular más conocida incluye grupos con distancia corta entre elementos como también espacios de área densos.

El Clustering puede ser formulado como un problema multi-objetivo de optimización, debido a que es muy importante el seteo de parámetros iniciales incluyendo la función de distancia a utilizar, los umbrales de densidad o el número esperado de clusters. Todo esto varía según la intención por la cual se utiliza dicha tarea.

Podría decirse que esta tarea no es automática, sino un proceso iterativo de descubrir cuáles son los parámetros que mejor optimizan la técnica, ya que se trata de ensayo y error. Frecuentemente es necesario modificar la forma de procesar la información y los parámetros de los modelos hasta poder encontrar el resultado deseado en términos de negocio.

3.3. Objetivo de la solución

3.3.1. Valor agregado al cliente

Como mencionamos anteriormente en la sección 2. Contexto y problema, nuestro cliente nos mencionó que la solución actual ofrece recomendaciones al usuario de productos estrechamente relacionados, en base al historial de productos adquiridos y no lo considera como parte de un conjunto de usuarios que comparten intereses.

Además, según lo manifestado, la solución actual es una primera aproximación un tanto simple y básica que aborda el problema de generar recomendaciones a usuarios, pero que no es completa desde el punto de vista de la experiencia de usuario.

Nuestra solución buscó ir un paso más adelante en esta problemática, pudiendo recomendar otros productos a los usuarios en base a un conjunto de intereses de usuarios similares.

Luego de muchas iteraciones con el cliente, con el propósito de definir el concepto de perfil de usuario, llegamos a que el mismo (en el contexto de este proyecto) estará determinado por su comportamiento de compra dentro de Mercado Libre.

Como mencionamos en la descripción del contexto, los productos publicados para la venta en Mercado Libre están asociados con una categoría. Estas categorías son puntos clave para entender el comportamiento de los usuarios y elaborar su perfil, por lo que, al agrupar usuarios por intereses en categorías similares, podemos recomendarles productos de categorías preferidas por sus vecinos cercanos.

En la siguiente imagen explicaremos el objetivo deseado.

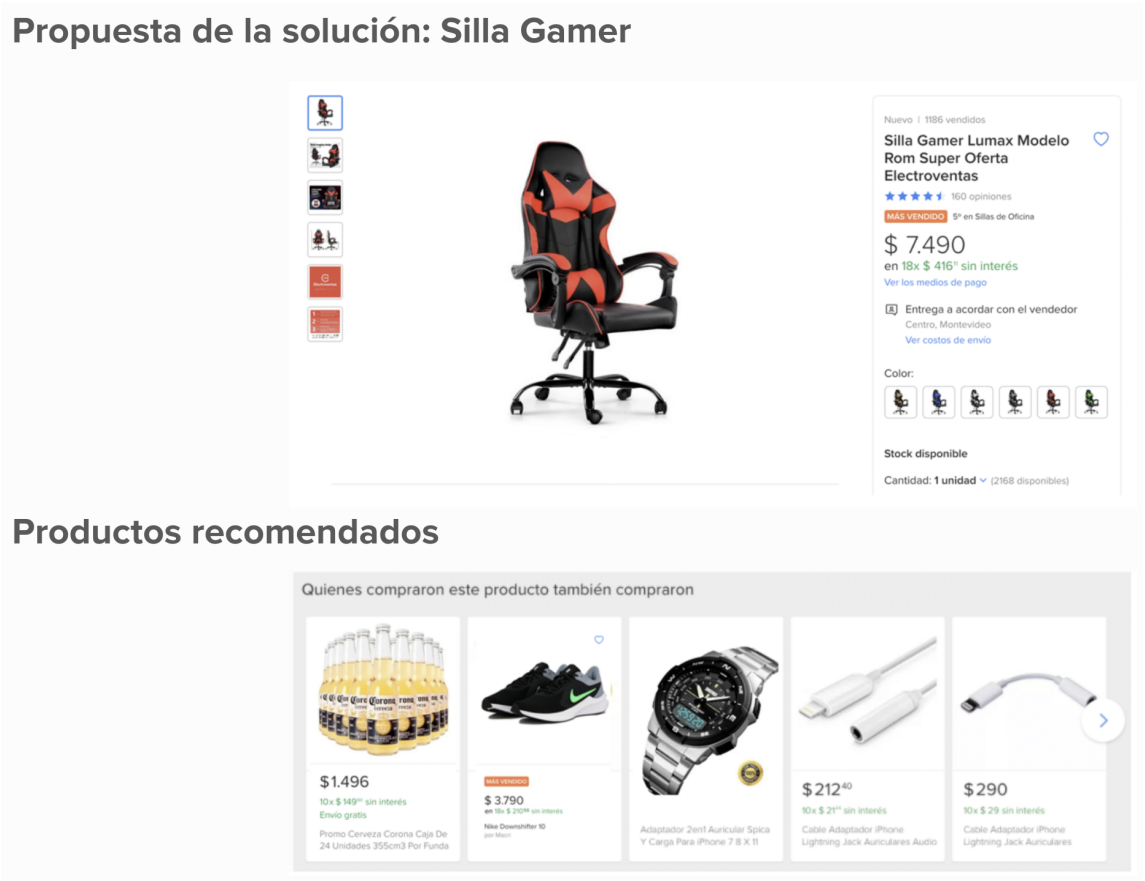


Figura 3.1: Productos recomendados con la nueva solución.

Aplicando lo mencionado anteriormente, si entendemos que el usuario actual califica para un perfil, por ejemplo Gamer, le recomendamos productos que sean de interés de un conjunto de usuarios Gamers, mejorando el modelo actual al ofrecer productos más adaptados al usuario y no necesariamente relacionados a compras realizadas por el mismo anteriormente. Aquí radica el valor más importante que aporta nuestra nueva solución.

Gracias a este enfoque apuntamos a recomendar, por ejemplo, un pack de botellas de cerveza, zapatillas o relojes a una persona considerada como “Gamer” (entendemos que son los artículos más comprados por los “Gamers”).

3.4. Requerimientos

En esta sección describiremos brevemente los requerimientos funcionales y no funcionales de la solución según la clasificación IEEE[7]. La lista completa de los requerimientos priorizados en formato de historia de usuario se especifican en el Anexo 1 - Product Backlog.

3.4.1. Requerimientos funcionales

Identificación de perfiles de usuario

Para nuestro cliente es muy importante poder entender el comportamiento de los usuarios en la plataforma, teniendo en cuenta los retos que esto presenta ya que no a todos les interesan los mismos productos.

Debido a la gran cantidad de usuarios existentes, se optó por un enfoque sobre grupos de usuarios, por esta razón poder agruparlos es fundamental para el negocio como nos comentó nuestro cliente.

Por esta razón se optó por una estrategia de agrupación para la conformación de los perfiles de usuario, que definimos junto con el cliente el concepto de que dos usuarios pertenecen al mismo perfil si y sólo si, estos tuvieron compras en las mismas categorías en cierta ventana de tiempo.

Con este enfoque, se buscó poder trabajar de forma individual con cada grupo, pudiendo así tener un resultado más eficiente teniendo en cuenta que no todos los usuarios tienen los mismos intereses como comentamos anteriormente.

Manejo de roles

Debido a que detectamos la actividad de diferentes actores del sistema, debíamos considerar distintas políticas de acceso para determinadas funcionalidades del mismo, por esta razón definimos varios tipos de acceso según el rol del usuario de negocio que ingrese al mismo y así restringir las funcionalidades específicas a quienes corresponden.

Existen tres roles de usuario en el sistema, que detallaremos a continuación:

- **Usuario manager (Manager User):** Es el rol que dispone de mayor nivel de permisos dentro de la aplicación, por lo que puede acceder a todas las funcionalidades.
- **Usuario Analista de Datos (Data Analyst):** Es el rol que permite a los usuarios analistas de datos ver únicamente la sección de perfiles.
- **Usuario Operador de Marketing (Marketing Operator):** Es el rol que permite a los usuarios encargados de Marketing acceder a la información sobre las campañas publicitarias enviadas a los usuarios.

Envío de recomendaciones a usuarios

Parte de la solución integral de todo el sistema fue poder brindarle la propuesta de valor a todos los usuarios involucrados y por eso la etapa final fue el envío de las recomendaciones.

Si bien la generación de las recomendaciones fue parte del desarrollo, el envío de las mismas se completó gracias a la interacción con otro sistema (denominado Hermes).

Visualización de información de envíos de publicidad

Para los usuarios con rol de Manager y Operador de Marketing es importante poder comprender la información referente al envío de campañas de marketing enviadas en determinados rangos de fechas, para entender principalmente cuántos usuarios fueron afectados.

3.4.2. Requerimientos no funcionales

En esta sección describimos los requerimientos no funcionales que restringieron a la solución desarrollada en forma global.

Interoperabilidad

Los módulos de la solución desarrollada debían interoperar dentro del ecosistema interno conformado por todos los subsistemas de Mercado Libre.

Confidencialidad

Se debió respetar la privacidad de datos de los clientes de la compañía, ocultando información sensible de los mismos como por ejemplo nombre, documento de identidad, etc, además de información de seguridad de accesos a recursos.

Stack tecnológico

Debido a que estuvimos trabajando en el entorno de desarrollo de Mercado Libre, fue obligatorio apegarse al stack tecnológico utilizado por la empresa. Para poder seguir con este lineamiento el equipo trabajó dentro de la plataforma de Fury para cumplir con este punto. En la sección 3.5.1. Suite Fury, se detalla la herramienta.

Utilización de los recursos de infraestructura de la compañía

El proyecto debió funcionar sobre la infraestructura de cloud de Mercado Libre y debió ser creado mediante herramientas internas de la compañía.

Look and feel

Se debió utilizar el layout de la compañía para las aplicaciones de frontend que se desarrollaron.

Acceso a la aplicación

La herramienta debió ser accesible para managers, data scientist y operadores de marketing de Mercado Libre, estando disponible dentro de la red interna privada de la empresa.

Estándares de calidad

La solución debió cumplir con los estándares de calidad estipulados por la empresa, en el proceso automatizado de integración contínua, como por ejemplo requisitos de coverage, checks de sintaxis, estándares de codificación, etc.

Especificaciones técnicas

El módulo de identificación de perfiles requirió funcionar sobre una infraestructura determinada con especificaciones de CPU, cantidad de memoria RAM, acorde al volumen de datos procesados.

3.5. Tecnologías aplicadas

A continuación describimos las tecnologías utilizadas en el desarrollo de la solución.

3.5.1. Suite Fury

Fury es una suite de herramientas provista y construida por el equipo de desarrolladores de Mercado Libre con el fin de facilitar la creación, mantenimiento y despliegue de cualquier tipo de aplicación utilizada dentro del ecosistema de la empresa.

Esta suite provee un set de herramientas que nos facilita todas las operaciones típicas del ciclo de vida de una pieza de software, como también la creación a demanda de servicios de terceros para integrarlos con nuestras aplicaciones, como podría ser una base de datos, un repositorio de almacenamiento externo, etc.

En la siguiente imagen mostramos un fragmento de la pantalla principal de nuestra aplicación de backend en Fury. En el panel izquierdo disponemos de un conjunto de acciones que podemos realizar sobre ésta, ya sea para generar una nueva versión y así deployar en un ambiente de test o productivo, configuraciones sobre la infraestructura en donde se aloja nuestra aplicación, entre otras más.

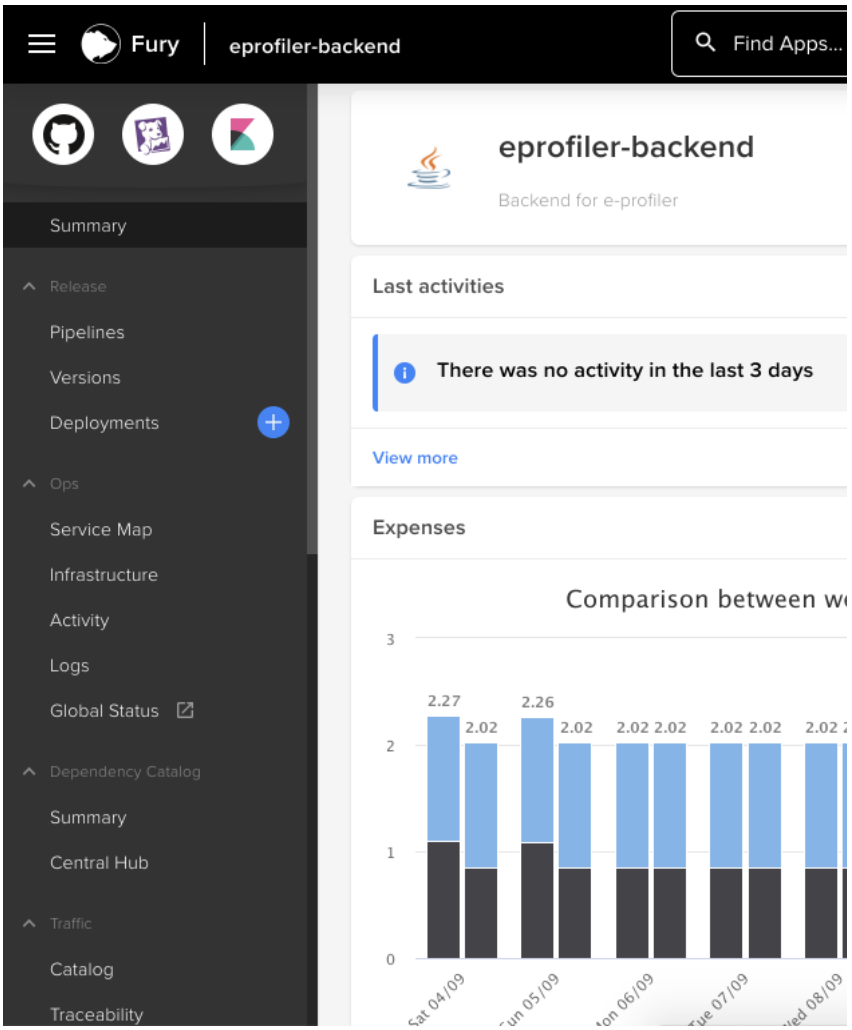


Figura 3.2: Presentación del frontend de Fury para la aplicación del backend.

A continuación mostramos un fragmento de la pantalla en el que se nos ofrece integración con una base de datos. Desde esta página pudimos fácilmente solicitar una base de datos y vincularla con nuestra aplicación en forma sencilla.

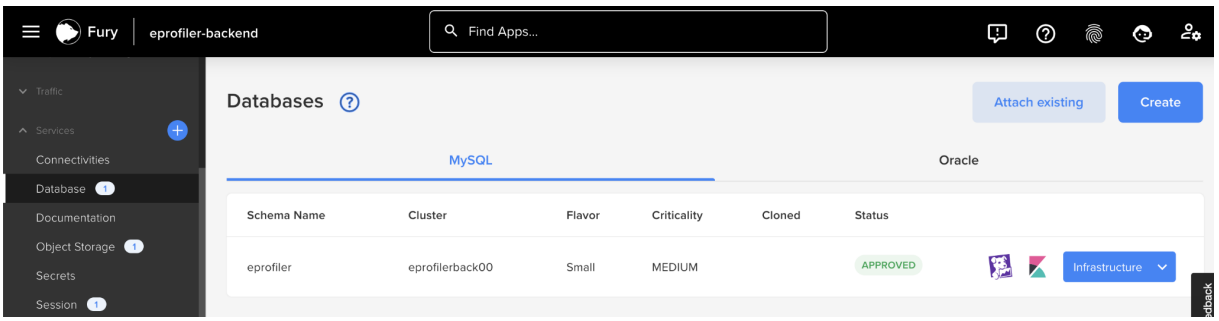


Figura 3.3: Sección de servicios de databases de Fury.

Si bien hay una amplia gama de opciones a configurar, existen otras que no son permitidas desde el menú, como por ejemplo la configuración del proveedor de servicios a utilizar (por ejemplo Amazon o Google Cloud) entre otras. En caso de ser necesario es posible contactar al equipo que administra la herramienta y recibir soporte mediante la creación de un ticket.

Cuando creamos la aplicación dentro de Fury esta nos permitió elegir un template de solución con varias dependencias ya instaladas que nos permitieron agilizar el desarrollo.

Nuestro trabajo, concretamente, consistió en desarrollar las aplicaciones de frontend, backend y de data mining sobre ese template provisto por la herramienta, que ya incluía un conjunto de librerías, además de integraciones que nos facilitaron las puestas en producción y el monitoreo.

3.5.2. Aplicaciones que componen la solución

La solución desarrollada en este proyecto consiste de tres tipos de aplicaciones, una Fury Data App, un módulo de backend y otro de frontend.

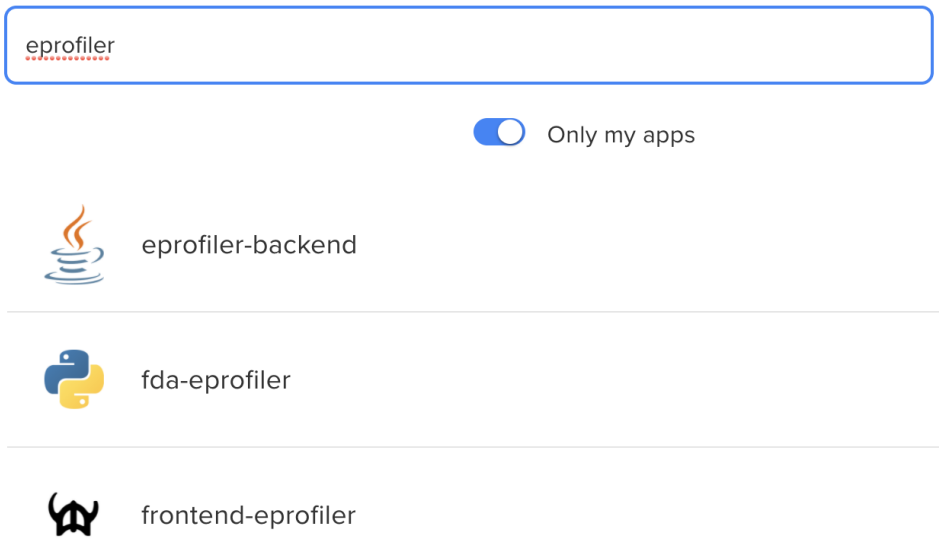


Figura 3.4: Aplicaciones creadas dentro de la suite fury para el proyecto.

Aplicación Fury Data App (FDA)

Este tipo de aplicación fue creada con el template correspondiente dentro de Fury, el cual permite que cualquier persona dentro de Mercado Libre pueda construir, mantener y desplegar una solución de Data Mining. Estas aplicaciones se diferencian de las tradicionales de frontend y backend, debido a que el template generado es sobre el lenguaje python. Dentro de esta aplicación de tipo FDA se generan dos placeholders particulares para cada etapa de la aplicación, las cuales detallamos a continuación.

ETL - Extraction/Transformation/Load

Esta etapa permite alojar el código relacionado a la extracción de datos para nuestro proyecto de Data Mining, conectándose con diversos orígenes de datos y realizando el proceso requerido para los algoritmos de identificación de perfiles y recomendaciones. Este proceso comienza con consultas SQL, transformaciones de datos y finalmente la carga de los resultados para luego reutilizarlos.

Estos resultados son almacenados por Fury dentro de contenedores especiales en proveedores en la nube. Esto nos permite correr aisladamente cada una de las etapas en el sistema de forma independiente de las demás.

Train - Model

Aquí reside la lógica de nuestros algoritmos o técnicas de modelling para los proyectos de Data Mining. Este contenedor de código es el responsable de alojar todo el código relacionado al procesamiento y manejo del modelo de datos que son obtenidos desde el lugar de almacenamiento del paso anterior.

Como mencionamos anteriormente, tanto el ETL como el Train ejecutan en infraestructuras diferentes, pero dentro del ambiente de la suite Fury. Esta herramienta es la encargada de abstraer y ocultar a nosotros como desarrolladores la lógica y las configuraciones de dónde se encuentran alojados los puntos de almacenamiento de datos entre otros (por ejemplo contenedores de Amazon S3 o Google Cloud Platform storage).

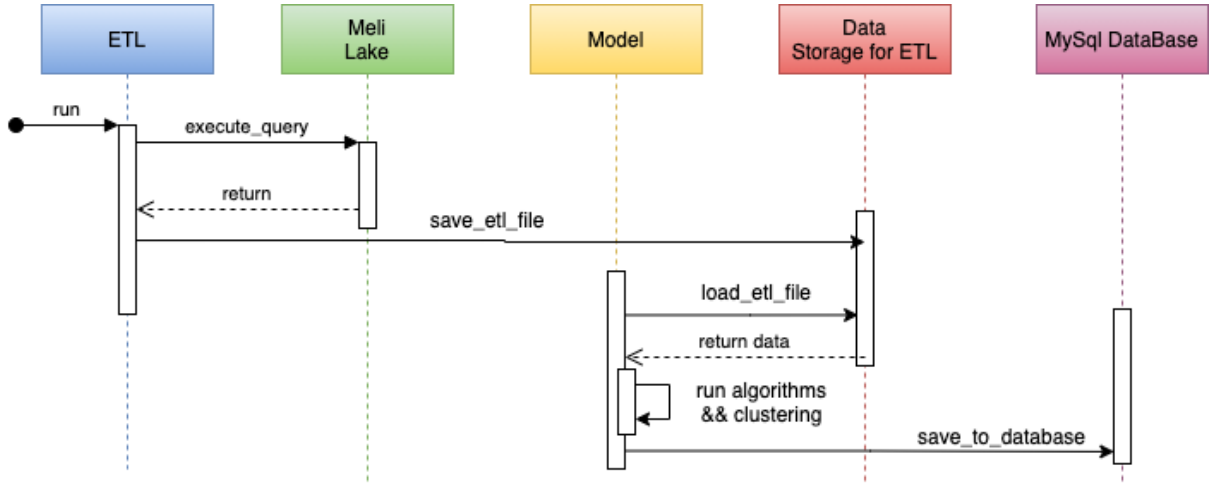


Figura 3.5: Diagrama de secuencia de funcionamiento de aplicación FDA.

Estas dos etapas son ejecutadas de forma independiente a demanda desde la consola principal de Fury o en forma automatizada. Una vez ejecutado el proceso pudimos almacenar el resultado (los usuarios y categorías deseadas) en un medio de almacenamiento en la nube para luego volver a utilizarlo a posteriori. Esto facilita el reuso, debido a que permite volver a obtener el resultado sin la necesidad de ejecutar nuevamente el ETL, dándonos las ventajas de ahorro de tiempo y procesamiento.

Una vez con el resultado del ETL almacenado, desde la etapa de Train pudimos obtener dicho conjunto de datos para aplicar los algoritmos de Clustering, Collaborative Filtering, entre otros y así arribar al resultado final que luego mostramos al usuario en el frontend.

Dado que nuestro proceso de Train (entrenamiento del modelo) se encargó de procesar la lógica y el resultado producido por este era la entrada para lo que íbamos a mostrarle a los usuarios a través del frontend, se decidió persistir en una base de datos MySQL debido a que se trabajaba con información tabular y además de ayudarnos a resolver el problema nos permitió rápidamente poder obtenerla desde la aplicación backend para presentarla al frontend.

Aplicación Backend

El segundo tipo de aplicación que utilizamos en esta solución fue una de tipo Backend. Este tipo de aplicación también fue creada sobre la suite Fury, la cual nos

permite construir rápidamente un backend web sobre varios lenguajes de programación. Esta fue fundamental para alojar parte de la lógica de negocio para popular los datos del frontend del producto entregado. En nuestro caso desarrollamos el backend del proyecto en el lenguaje Java integrado con el framework Spring.

Aplicación Frontend

El tercer y último tipo de aplicación utilizado para construir la solución del proyecto fue una aplicación de tipo frontend, creada sobre la suite Fury. Esta aplicación contiene el código necesario para implementar la capa de presentación de datos para interactuar con los distintos tipos de usuarios de marketing de Mercado Libre que hacen uso de la misma. El stack de tecnologías utilizado fue Node JS y React, que corresponde al estándar de la compañía.

3.6. Arquitectura de la solución

En esta sección describimos el diseño arquitectónico a través de un conjunto de vistas propuestas por el modelo 4+1 [6].

De estas se desprenden distintas responsabilidades acorde a cada nivel para su posterior explicación. Cabe destacar que aquí presentaremos la solución que construimos, además de las decisiones de diseño que se tomaron durante el proceso.

3.6.1. Vista de componentes

A continuación detallaremos nuestra solución con una vista de componentes explicando cada uno de ellos.

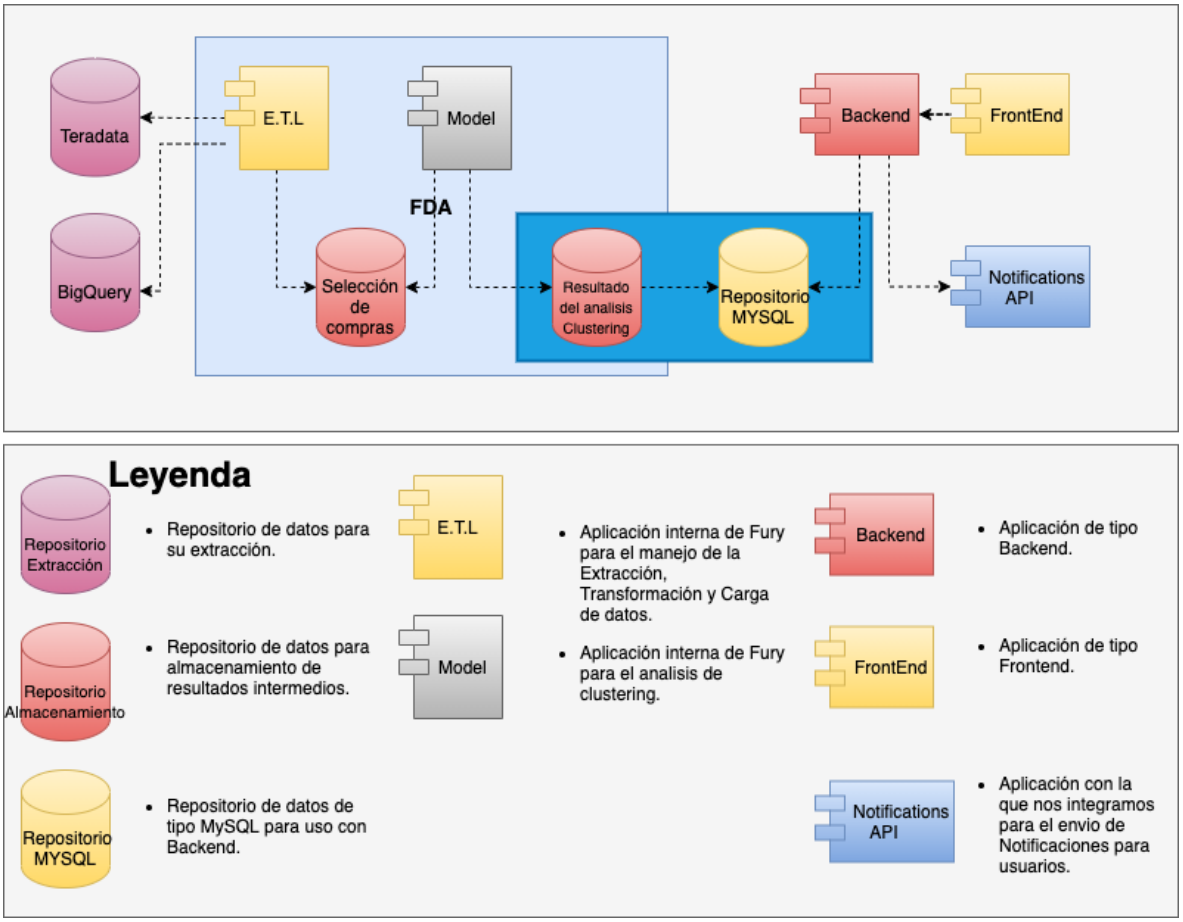


Figura 3.6: Vista de componentes de la solución.

Descripción de componentes

En primer lugar, Teradata y BigQuery corresponden a orígenes de datos utilizados en nuestra solución. Estos componentes son utilizados para extraer la información de compras de usuarios y categorías respectivamente.

El componente de ETL, como ya mencionamos, tiene las responsabilidades principales de extraer datos de los orígenes mencionados, procesarlos y luego almacenarlos en un repositorio en la nube provisto por Fury.

Model es el encargado de almacenar la información resultante del entrenamiento del modelo. El entrenamiento consiste en ejecutar los algoritmos de identificación de

perfiles y recomendaciones y almacenar el resultado en un almacenamiento intermedio (dentro de la Fury Data App).

Utilizamos un almacenamiento de datos relacional (base de datos de tipo MySQL), para almacenar los resultados en forma tabular. La exportación de datos hacia el backend la realizamos con el objetivo de facilitar la posterior la consulta desde el mismo para luego servir los datos al frontend.

Con el objetivo de enviar las recomendaciones de productos a los usuarios, desde el componente de backend fue necesario integrarse con el de Notificaciones API (Hermes) a través de mensajes HTTP.

3.6.2. Vista de despliegue

Debido a la gran complejidad que maneja internamente Fury, de forma de poder proveer a los distintos desarrolladores de la plataforma construir, mantener y desplegar aplicaciones, intentaremos dar una breve explicación sobre su funcionamiento.

Esta suite es la encargada de gestionar el ciclo de vida de nuestras aplicaciones, como también facilitar varias etapas durante el ciclo de desarrollo. Además existen diversas decisiones que sustentan esta suite pero razones de alcance, al tratarse de un proyecto académico vamos a resumir su funcionamiento a un alto nivel.

Debido a que Fury se encarga de gestionar la creación de varios componentes, muchas veces es deseable desacoplar los proveedores de servicios y también por razones de seguridad y escalamiento, es deseable ubicarlos en distintos nodos.

Por esta razón Fury crea nodos o espacios de ejecución en proveedores cloud con el fin de poder garantizar un máximo desempeño y escalabilidad para cada una de estas. Las comunicaciones entre los nodos muchas veces son canales internos que la suite ofrece pero oculta su implementación a los distintos clientes.

Las comunicaciones con el exterior son realizadas mediante peticiones del tipo HTTP en donde mediante mensajes del tipo GET, POST, PUT, DELETE se puede acceder a los distintos recursos expuestos.

A continuación presentamos el diagrama de despliegue de la solución.

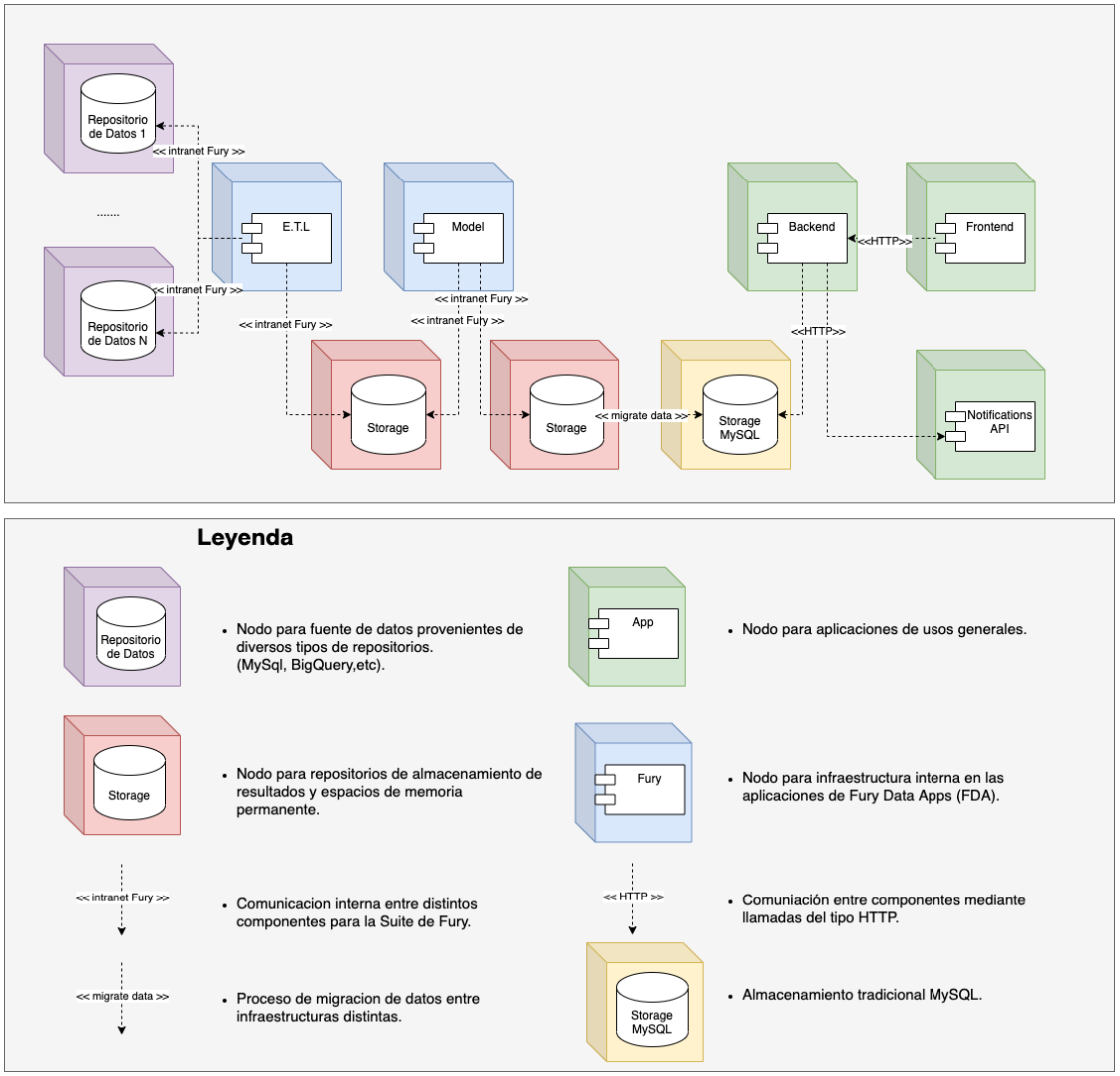


Figura 3.7: Diagrama de despliegue.

3.6.3. Decisiones de diseño

A continuación explicaremos las decisiones de diseño tomadas para lograr un balance entre las necesidades del cliente y los requerimientos no funcionales que condicionan la arquitectura de nuestro sistema.

Todos los artefactos (bases de datos, aplicaciones, módulos de FDA) fueron desarrollados utilizando el stack tecnológico provisto por la compañía, respetando así el requerimiento no funcional de stack tecnológico explicado en la sección de requerimientos no funcionales del Anexo 1: Product Backlog.

Componente de ETL

Como explicamos anteriormente, el componente de ETL es el artefacto responsable de la ingesta de datos para el sistema. Si bien en etapas iniciales disponíamos de un laboratorio de menores recursos cuando comenzamos a trabajar con las consultas que abarcaban un año de datos, requerimos escalar la infraestructura que nos permitió ejecutar y almacenar los registros obtenidos.

Durante esta etapa fue necesario la obtención de los usuarios que fueron utilizados en los algoritmos de collaborative filtering y una vez obtenidos, junto con las categorías en donde habían realizado las compras, se procedió a almacenar el resultado en un almacenamiento externo (a través de la herramienta Fury).

Dicha estrategia nos permitió disponer de un lugar específico en donde se almacenaban los resultados de los distintos ETL, teniendo en cuenta que era necesario generar un punto de convergencia entre el resultado de la ejecución del ETL y la etapa de Train (que detallaremos a continuación), responsable de obtener dichos datos y ejecutar la lógica de clustering entre otras.

Decidimos conveniente el uso de variables globales llamadas “secrets” o secretos de su traducción al español, para el manejo de las credenciales necesarias para la conexión hacia las base de datos (usuario y contraseña) que por razones de seguridad y mantenimiento se almacenaron de esta forma. Esto permite que nuestro diseño arquitectónico cumpla con el requerimiento no funcional de confidencialidad de datos (RNF2).

El manejo de los secretos lo realizamos a través de Fury, quien mantiene los valores (en este caso de usuario y password de la conexión) ocultos por seguridad y permite accederlos desde dentro de las aplicaciones en forma segura.

```
user = get_secret("MELI_USER")
passwd = get_secret("MELI_PASS")
```

```
tera = ConnTeradata(user,passwd,auth_method=Authentication.LDAP)
```

Figura 3.8: Uso de secretos y del conector al origen de datos Teradata.

Fury nos provee una librería de conectores a los distintos orígenes de datos, por lo que realizamos la consulta a Teradata como mostramos a continuación. El hecho de externalizar la configuración de la consulta nos permitió maximizar la modificabilidad de la solución, de forma tal que en caso de necesitar cambiarla, el código de nuestra aplicación no se vería afectado.

```
compras_de_usuarios = tera.execute_response(QUERY_COMPRAS_USUARIOS)
```

Figura 3.9: Ejecución de consulta mediante conector provisto por Fury.

Componente Model

El componente Model, como ya mencionamos, fue el encargado de contener la lógica responsable del procesamiento de datos utilizando varios algoritmos, como también técnicas para procesar la información de usuarios, compras y categorías obtenidos en la etapa de ETL, produciendo así la salida esperada.

A continuación detallamos las técnicas, algoritmos y puntos más destacados.

Proceso de migrado de datos

Debido a que Fury dispone de un flujo determinado para la entrada y salida de datos para las aplicaciones de tipo de Data Mining, nos vimos obligados a tomar una decisión de diseño importante para poder obtener la información por parte del análisis de clustering.

Debido a que nosotros, a través del análisis ya generamos un resultado fijo dado una entrada determinada, este resultado estaba en condiciones de ser consumido de forma directa por nuestro backend. De esta forma logramos utilizar menos recursos de memoria, ya que gran parte del resultado no era necesario disponerlo en memoria, provocando así una reducción en los costos considerable, lo cual fue validado por nuestro cliente.

Por esta razón y una vez producido el resultado del análisis de clustering, a través de un proceso de migración (otro servicio provisto por la suite de Fury), fuimos capaces de migrar la información de una infraestructura costosa a una más económica.

Clustering

La solución contó con un análisis de Clustering[8] dentro de la etapa de Modelling. Gracias al algoritmo KMeans[9], se pudo llegar a un conjunto de grupos de usuarios o perfiles. Dicho algoritmo fue escogido por recomendación del Líder Técnico, ya que no disponíamos de mucha experiencia en algoritmos de este tipo para aplicaciones similares. Además existe una gran cantidad de documentación disponible, teniendo en cuenta que es de gran aceptación en el mundo académico.

Collaborative filtering

Dicho método[10] consiste en conformar una matriz en donde nuestra filas estaban constituidos por usuarios, mientras que las columnas corresponden a las categorías en donde los usuarios realizaron compras.

En nuestra solución decidimos utilizar el tipo híbrido de filtrado colaborativo, debido a que operamos con un gran volumen de datos dispuestos en una matriz dispersa.

La técnica de neighbor-based collaborative filtering nos permitió cruzar información de usuarios y categorías de productos que hayan adquirido y de esta forma poder determinar similitudes entre usuarios. Esto nos permitió agruparlos y recomendarle productos en los que otros usuarios similares estuvieron interesados.

Esta matriz contaba con una gran cantidad de posiciones vacías, debido a que bajo la premisa de que no todos los usuarios compran los mismos productos, consecuentemente se iba a cumplir que no todas las posiciones tenían un valor definido, lo cual nos originó el problema de manejar una matriz que no podía almacenarse en memoria. Este problema es característico de esta técnica y lo resolvimos como comentamos en el siguiente párrafo.

Para poder mitigar este problema y así poder mejorar la performance de los algoritmos, se aplicaron técnicas que nos permiten comprimir la matriz de usuarios y compras, con el fin de trabajar con una de dimensiones menores pero equivalente a la primera. Esto nos permitió acelerar el tiempo de procesamiento y aumentar la eficiencia del proceso.

Para llevar a cabo el procesamiento de las matrices se utilizó la librería de Scipy. Gracias a esta librería se pudo contar con un amplio set de funciones, como por ejemplo comprimir una matriz dispersa a otra equivalente de menor tamaño. A continuación mostramos un fragmento de código con el uso de esta y su resultado.

```
#compress sparse matrix
X_sparse = csr_matrix(data_from_customers)

X_sparse_svd = svd.fit_transform(X_sparse)
```

Figura 3.10: Uso de Scipy - csr_matrix.

El resultado de la compresión de la matriz dispersa, es otra matriz equivalente que contiene únicamente información sobre las posiciones no nulas para donde existen valores en la matriz original, es decir se comprime considerablemente el tamaño de una matriz y en algunos casos dicha reducción puede ser casi la mitad de tamaño.

Una vez reducida la matriz procedimos a aplicar otra técnica de reducción de dimensiones sobre ella. Esto nos permitió complementar con la performance del cálculo ya que dicha matriz es de menor tamaño aún pero equivalente.

Tener en cuenta que la primera técnica no redujo la dimensión, sino que solamente la comprimió, manteniendo las mismas proporciones pero en lugar de almacenar en memoria una matriz de dimensiones Clientes por Categorías de tamaño (matriz completa) únicamente se almacena información sobre las posiciones en donde dicha matriz tiene valor definido.

Para esta segunda técnica nos apoyamos en la librería de Sklearn[11] llamada TruncatedSVD[12] que nos permitió lograr dicho cometido.

Justificación en la elección de los algoritmos

Nearest-Neighbors y K-means son algoritmos usados tanto a nivel educativo como también a nivel empresarial en diversos tipos de proyectos. Gracias a ellos pudimos arribar a la solución propuesta. Ambos fueron resultado de una extensa investigación para el equipo ya que no se contaba con experiencia previa.

Si bien fueron presentados a nuestro Product Owner técnico para su validación en su uso, además fueron fáciles de implementar gracias a la amplia documentación disponible en la web.

Nuestro primer spike contó con la utilización de ambos algoritmos que una vez obtenidos resultados fueron aprobados por nuestro Product Owner y esto nos permitió seguir adelante. Cabe destacar que en nuestro proceso de investigación aprendimos como otras empresas de rubros diferentes los utilizaban con los mismos fines y por esta razón decidimos una vez más apoyarnos para la elección final.

Fury Data Apps (FDA) y su aplicación

FDA es el framework construido dentro del equipo de Mercado Libre única y exclusivamente como una herramienta de uso interno para la construcción, mantenimiento y despliegue de cualquier solución de tipo Machine Learning que permite trabajar con grandes volúmenes de datos.

Además de ser la única herramienta bien comprendida por todos los equipos de desarrollo, se dispone de un equipo que se encarga de administrar y liberar nuevas versiones de la misma con nuevas funcionalidades para facilitarnos el trabajo.

Gracias a FDA, pudimos completar la identificación de perfiles de usuario y el módulo de productos recomendados. Más conceptualmente, nos permitió llevar adelante un proyecto de data mining pudiendo encapsular su lógica, pudiendo así operar con grandes volúmenes de información, almacenarla y utilizarla a demanda. Sin esta herramienta hubiera nos sido difícil poder completar las funcionalidades mencionadas, debido a las necesidades de infraestructura que se requieren.

Aplicación e-Profiler backend

La aplicación de backend nos permitió trabajar con los datos obtenidos del análisis de perfiles y recomendaciones, además de realizar operaciones sobre estos y centralizar la lógica para manipularlos. Otra responsabilidad que cumple este backend es la integración con el módulo de notificaciones (sistema denominado Hermes) para envío de recomendaciones de productos a usuarios de Mercado Libre.

Cumpliendo con el requerimiento no funcional de interoperabilidad entre los distintos sistemas del ecosistema de Mercado Libre, nuestro backend fue diseñado con el fin de interconectar nuestro set de datos proveniente de la solución de Data Mining con los módulos de Frontend y Notificaciones. Toda las llamadas entre los sistemas son a través de una red interna dentro de la infraestructura de Fury y gracias a la conexión con VPN pudimos garantizar la seguridad en el tráfico, pudiendo así también cumplir con el requerimiento no funcional de confidencialidad (RNF2) explicado en el Anexo 1: Product Backlog.

Aplicación e-Profiler frontend

La aplicación de tipo frontend nos permitió construir la interfaz gráfica para ser utilizada por el personal de marketing de Mercado Libre para la visualización de datos. Esta aplicación se comunica internamente con el backend descrito anteriormente para la obtención de datos de perfiles y notificaciones.

Con el objetivo de validar los permisos de cada uno de los usuarios del sistema, implementamos una lógica que valida sus roles, brindando acceso solamente a las secciones permitidas para cada rol. De esta forma cumplimos con el requerimiento funcional de manejo de roles.

Envío de recomendaciones : Notifications API

Es importante destacar que para cumplir con este requerimiento, parte del desarrollo consistió en la integración con una API de Notifications (sistema Hermes) previamente construida por otro equipo de desarrolladores de Mercado Libre facilitando en gran medida el esfuerzo necesario para completar esta funcionalidad.

Dicha API expone métodos específicos que mediante el posteo de un mensaje de tipo HTTP[13], pudimos generar notificaciones de diversos tipos dirigidas a usuarios reales de la plataforma (por el momento sólo enviamos emails promocionales).

Todo este trabajo de lógica fue desarrollado dentro de nuestra aplicación de backend, en la cual pudimos iterar para cada usuario de cada perfil pudiendo así enviar los productos recomendados correspondientes.

Por razones académicas y limitaciones por ser un proyecto nuevo, no se realizó ningún envío de mail real a los usuarios de forma automática. Si bien fue validado el resultado por parte del cliente, realizar la prueba completa con el envío de dichas recomendaciones quedará pendiente para una futura iteración.

4. Metodología

En esta sección describimos las metodologías que utilizamos en el proyecto, además de estudios de alternativas que tuvimos en cuenta para elegir las más apropiadas para cada fase del mismo y así explotar al máximo las ventajas que cada una de ellas nos ofrece.

Por esta razón definimos que la mejor decisión sería la de dedicar un tiempo para investigación (etapa de Discovery) antes de dedicarnos completamente a desarrollar la solución y luego comprometimos entregas con el cliente que le fuimos realizando, etapa que denominamos Delivery. Ambas las explicaremos a continuación.

4.2. Fase de discovery

Al comienzo del proyecto, nuestro cliente nos brindó una idea de alto nivel, la cual debimos bajar a tierra y disminuir la gran incertidumbre que existía.

Por esta razón nos basamos en una estrategia liviana utilizando un tablero tipo Kanban[14] en donde se detalló la lista de tareas necesarias de alto nivel. Debido a la gran incertidumbre presentada durante esta etapa se definió la construcción de bosquejos (presentados en el Anexo 2: Mockups de frontend) y pruebas de concepto para validar con el cliente la solución final.

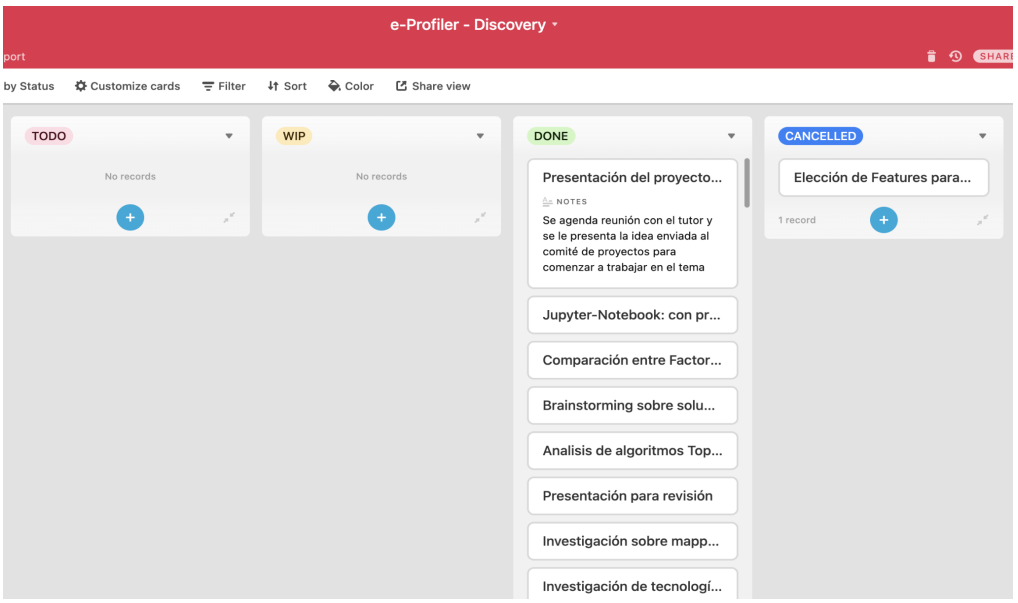


Figura 4.1: Tablero basado en Kanban utilizado para la organización de tareas.

Como equipo evaluamos la utilización de otras metodologías (por ejemplo Scrum), pero debido a la gran carga de ceremonias formales que requiere, además de otros esfuerzos de gestión requeridos, consideramos que era mejor manejar simplemente un tablero con una lista de tareas que nos permitió arribar a una solución validada en etapas posteriores con el cliente.

Cabe destacar que debido al reducido número de integrantes del equipo y la poca experiencia que teníamos sobre proyectos de este tipo, existían diversas dudas que hacían difícil poder generar un cronograma claro, como también la forma de trabajo a llevar adelante. Por esta razón utilizamos dicho tablero con varias tareas atómicas, lo que resultó más que suficiente para organizar nuestro trabajo.

Además como acordamos con el cliente no realizar entregas regulares durante la fase de discovery, esta forma nos permitió tener un flujo de tareas que se fueron completando constantemente y así acelerar el proceso de entendimiento del problema.

La fase de discovery, además de reducir la gran incertidumbre existente, nos permitió construir una prueba de concepto sobre el análisis de datos sobre la cual se continuó el desarrollo, principalmente para la etapa del proyecto correspondiente a Data Mining, ya que con el resto de los requerimientos nos sentíamos con un mayor nivel de seguridad.

4.3. Fase de delivery

Al igual que en la fase de discovery, en esta fase también manejamos distintas alternativas en cuanto a metodologías. Además de nuestra investigación, y considerando el feedback recibido por los Docentes en las revisiones parciales, decidimos que la mejor opción sería dividir nuestra solución en dos tipos de sub-proyectos: por un lado el de Data Science y por otro lado el que comprende las aplicaciones de frontend y backend. Esto se fundamenta en que la naturaleza del proyecto determinó el marco de trabajo y sabemos que existen metodologías que encajan mejor con cada tipo de proyecto, lo cual resultó beneficioso para el mismo.

Para las aplicaciones de frontend y backend ya se disponía de una lista de requerimientos mucho más clara, debido a que el equipo contaba con mayor experiencia para estos artefactos.

Por el otro lado para la solución de Data Mining, ya se contaba con algoritmos y técnicas validadas por el Líder técnico, como también una validación por parte del cliente. Las pruebas de concepto fueron de gran utilidad debido a que ayudaron a implementar una primera versión del algoritmo de clasificación de usuarios validada por el cliente.

4.3.1. Backend y Frontend

Para los proyectos de backend y frontend utilizamos metodologías de desarrollo de software ágiles, debido a que el equipo ya contaba con la experiencia en proyectos de este tipo.

Como primer nivel de elección de metodologías, decidimos optar por las ágiles (en contraposición a las tradicionales), debido a las siguientes razones:

1. Entorno cambiante: la empresa Mercado Libre tiene dentro de sus principios culturales estar en beta continuo, por lo que la realidad puede cambiar de un momento a otro. Por eso la metodología debe contemplar estos cambios en las realidades pudiendo adaptar el marco de trabajo con facilidad.

2. Incertidumbre en la definición de los requerimientos: los mismos se irán comprendiendo con mayor precisión a medida que avance el proyecto y no se tienen del todo claros al comienzo del mismo.

3. Comodidad y manejo de las metodologías ágiles: ambos integrantes del equipo trabajamos en la empresa a diario con este tipo de metodologías, lo que supone la ventaja de aplanar la curva de aprendizaje y avanzar más rápido con otros temas.

Utilizamos BDD (Business Driven Development) como marco de trabajo para el desarrollo de las aplicaciones de backend y frontend, debido a que nos permitió mapear claramente las necesidades de negocio en piezas de código que puedan probarse automáticamente y de esta forma validar que la implementación satisface

dichas necesidades. Cada requerimiento se relevó como historias de usuarios, donde cada una fue definida con su criterio de aceptación y de ahí sus pruebas, las cuales fueron escritas con Gherkin.

Gracias a Gherkin pudimos manejar un lenguaje común entre el equipo técnico y el del negocio, lo cual nos brindó muchas facilidades como también una gran claridad para el desarrollo.

En BDD utilizamos un tablero como mostramos a continuación en donde cada tarea pasa por las distintas etapas que propone la metodología, de forma tal que en cada una de ellas realizamos diferentes actividades que explicaremos brevemente más adelante.

Release Backlog

En esta etapa partíamos con una lista de requerimientos relevada en base a la aprobación de los mock ups presentados al cliente en las etapas iniciales del proyecto. De esta forma pudimos generar prácticamente la totalidad de los requerimientos y a medida que avanzábamos en los releases seleccionamos un conjunto a trabajar previa validación con el cliente.

Requirements Definition

En esta etapa definimos un conjunto de requerimientos que fueron analizados y ponderados con el cliente, junto con el análisis de los escenarios que luego formaron parte de los criterios de aceptación.

Test Cases Implementation

El objetivo de esta etapa consistió en escribir los casos de prueba en base a la narrativa de cada requerimiento analizado, junto con los escenarios para los casos de prueba a validar. Estos fueron codificados mediante la herramienta Gherkin utilizando una nomenclatura del tipo <given-when-then>, que representan de su traducción al español en: “Dado (given) una situación, cuando (when) sucede un evento entonces (then) su consecuencia”.

Application Implementation and Testing

Al ejecutar los test cases escritos en Gherkin y ver que fallan, entonces debe implementarse la aplicación hasta que todos se ejecuten correctamente y sean exitosos. Una vez que son exitosos, aseguramos que la implementación cumple con los requerimientos definidos y por lo tanto fue culminada la funcionalidad.

Integration Testing

En esta etapa realizamos pruebas de los componentes a nivel de integración con otros ya existentes en el sistema. De esta forma pudimos asegurar que los componentes eran capaces de integrarse correctamente.

Refactor

Etapa posterior a la implementación de la aplicación y el correcto desempeño de las pruebas, en la cual se realizaron mejoras sobre dicha implementación (por ejemplo, algoritmos más eficientes).

Done

Esta es la última etapa en donde si pasa todas las anteriores es considerada como completa.

A continuación presentamos un ejemplo del proceso que siguieron las tareas dentro del tablero de seguimiento del ciclo de BDD.

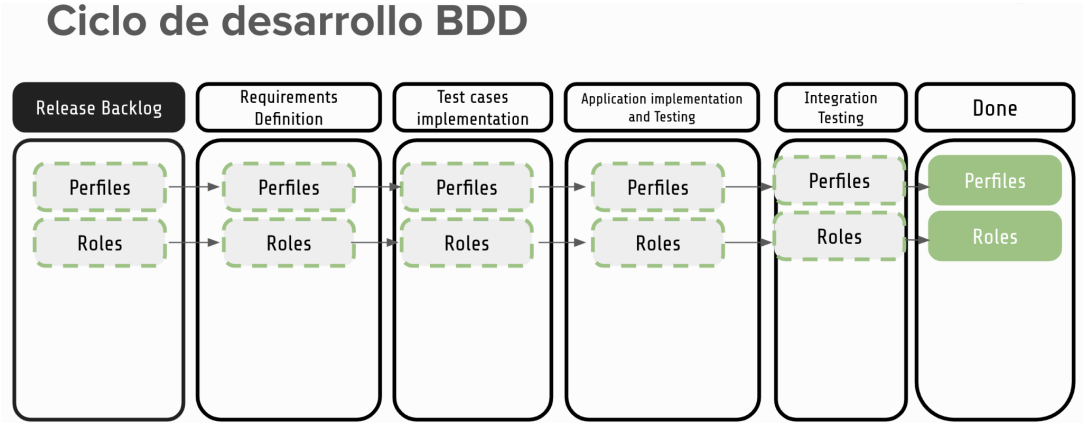


Figura 4.2: Ciclo de desarrollo en BDD.

Para cada parte de una user story creamos una tarjeta en Trello, en la que manejamos una descripción además del detalle del esfuerzo estimado y real en cada una de las etapas del ciclo de BDD. En la siguiente imagen mostramos un ejemplo de una tarjeta.

Figura 4.3: Descripción de tarea dentro de BDD.

4.3.2. Data Mining

Para el subproyecto de Data Mining, manejamos varias alternativas en cuanto a metodologías (en el anexo de metodologías las comentaremos con mayor nivel de detalle).

Debido a que no disponíamos de suficiente experiencia en proyectos de este tipo y conocimiento sobre cuál era la mejor metodología para desarrollar el producto, fue muy importante la ayuda de nuestro Technical Leader, Luis Argerich para la definición y asistencia.

Se evaluaron varias metodologías dentro de las que evaluamos Crisp-DM (Cross Industry Process for Data Mining), KDD (Knowledge Discovery in Databases) y SEMMA (Sample, Explore, Modify, Model, Assess), pero al final adaptamos Crisp-DM a nuestras posibilidades mediante la selección de un subconjunto de sus actividades debido a varios puntos que detallaremos a continuación:

- Es una metodología ampliamente utilizada en el mercado.

- Metodología sugerida por los Docentes revisores y expertos técnicos de la empresa.

- Metodología orientada a reducir la incertidumbre y a los objetivos de negocio.

De las principales razones que nos obligaron a tener que modificarla para que se ajustara a nuestro contexto fueron:

- La cantidad de miembros del equipo fue una gran restricción para poder respetar en tiempo y forma toda la cantidad de actividades que muchas veces se requiere para completar una conjunto de etapas.

- El orden de las actividades propuestas por la metodología es en forma de cascada, generando varios puntos de demora para poder completar un ciclo.

Si bien nos daba un marco de trabajo organizado para poder ir cumpliendo cada una de estas y así generar incrementos de software, para algunas etapas no respetamos el orden debido a que para varias iteraciones algunas etapas permanecen invariantes o para nosotros era necesario reforzar otras. En el Anexo 7: Evaluación de metodologías, se presenta una evaluación de metodologías de desarrollo y, dentro de ellas, para proyectos de data mining.

Nuestra forma de trabajo dentro del proyecto de data mining fue más simple que la detallada anteriormente para Backend/Frontend. Para este caso manejamos un tablero de Trello más sencillo que el anterior, y dado que utilizamos Crisp-DM como guía y no al pie de la letra, no consideramos necesario dar seguimiento en detalle de cada fase de la metodología.

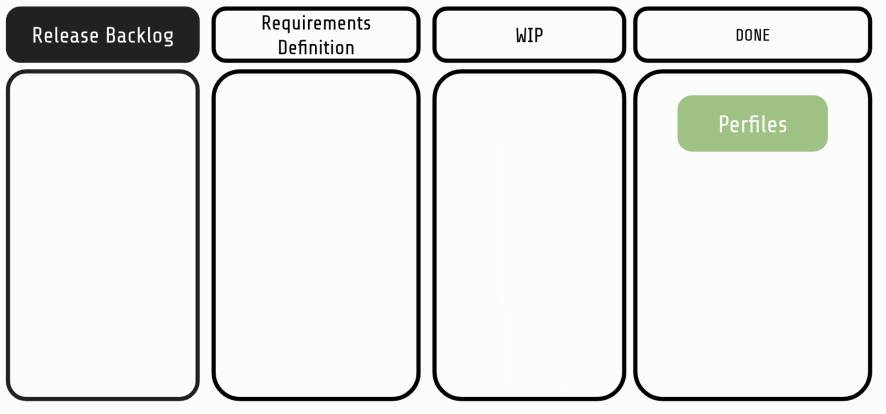


Figura 4.4: Ciclo de desarrollo de Crisp-DM.

Dentro de nuestro tablero de Trello manejamos las estimaciones y seguimiento de esfuerzo a nivel de las fases que propone la metodología Crisp-DM como detallamos en la imagen que se presenta a continuación.

Figura 4.5: Descripción de tarea dentro de Crisp-DM.

4.3.3. Resumen de marco de trabajo de la etapa

Nuestro marco de trabajo para ambos proyectos fue bajo las ceremonias explicadas anteriormente dentro de la metodología Scrum. Gracias a estas pudimos aplicar las etapas del desarrollo de software denominadas Plan-Do-Check que detallaremos a continuación. A continuación presentamos un diagrama de nuestro flujo de trabajo, en el que se muestran las etapas realizadas.

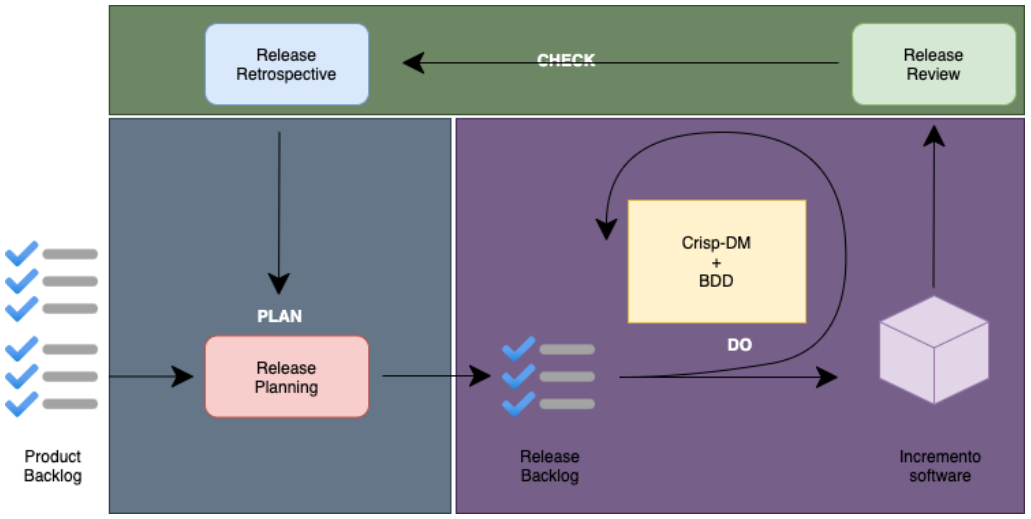


Figura 4.6: Esquema del flujo de trabajo.

PLAN

Release Planning

En la release planning seleccionamos un conjunto de user stories del product backlog y tareas a llevar a cabo en una unidad de tiempo que denominamos release. Para esta tarea tuvimos en cuenta las prioridades establecidas por el cliente para poder brindar el mayor valor al negocio lo antes posible.

Relevamiento de requerimientos

Si bien teníamos en claro cuáles eran las necesidades del cliente, una vez terminada la etapa de Discovery íbamos a tener consolidados claramente los requerimientos y gracias a las pruebas de concepto implementadas, pudimos confeccionar una lista con todos estos. Para visualizar el Product Backlog completo dirigirse al Anexo 1: Product Backlog.

Además de que disponíamos de una lista de requerimientos de alto nivel, para cada una de los releases pudimos confeccionar otra con un nivel de detalle más específico una vez comenzado el release. Lo realizamos de esta forma debido a que disponíamos de mayor información para profundizar con mayor claridad dentro de los requerimientos.

DO

Luego de confeccionado el Release Backlog comenzamos a desarrollar los requerimientos, organizando las tareas utilizando un tablero según la metodología como mencionamos anteriormente.

De la metodología Scrum utilizamos y respetamos la mayoría de las ceremonias, siendo estas la Release Planning, Release Retrospective y Release Review Meeting. Por otro lado, y dado que nuestras planificaciones para desarrollar fueron a nivel de release y no a nivel de Sprint, Scrum no se ajustaba de forma exacta a nuestras necesidades por lo que se incorporó un tablero de tipo Kanban dentro de cada release para mitigar este faltante.

Esto nos ayudó en gran medida puesto que las tareas no tenían un orden, pero si la totalidad de estas nos permitieron seguir adelante. Por lo anteriormente explicado, la metodología utilizada fue Scrum-Ban, que surge de la combinación entre Scrum y Kanban.

Este tablero nos permite poder avanzar con cada una de las tareas planificadas que fueron seleccionadas al momento de planificar cada uno de los releases. Una vez completada la totalidad de las tareas propuestas para dicho release, y entregamos al cliente, estuvimos en condiciones de pasar al próximo.

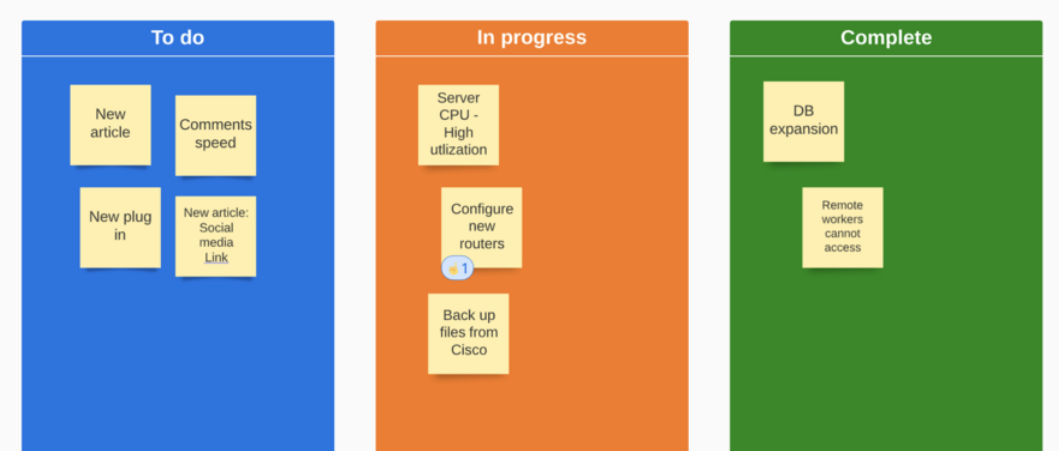


Figura 4.7: Tablero tipo Kanban.

One Piece Flow

One-piece flow significa que sólo una pieza es movida a la vez entre una etapa y la siguiente durante el desarrollo de las tareas y al igual que nuestra forma de trabajo nos permitió trabajar centrando nuestro esfuerzo en una única tarea por persona del equipo. Una vez completada esta la movíamos entre las distintas fases del tablero.

Mantener esta dinámica de trabajo nos permitió tener un flujo en paralelo óptimo, constante y en caso de que se presentara algún inconveniente era reportado al equipo con el fin de mitigarlo o bien re asignar la tarea.

CHECK

Para poder validar nuestro desarrollo tanto a nivel del proyecto como a nivel del equipo, utilizamos las ceremonias de Release Review y Release Retrospective como detallamos a continuación.

Gracias a estas, el equipo pudo reflexionar y trabajar en aspectos que hacen a la gestión del proyecto como también disponer de una instancia de validación y feedback con el cliente.

A continuación explicamos como trabajamos para cada una de las ceremonias propuestas por Scrum-Ban.

Release Review

Una vez finalizado el release, le pasamos informe de avance del proyecto al cliente, además de reunirnos brevemente, con el objetivo de relevar su feedback para la mejora contínua y relevar cambios que pudiesen haber surgido sobre lo entregado.

Release Retrospectives

Fueron reuniones que realizamos con el fin de evaluar el funcionamiento interno del equipo, entender de qué manera estuvimos ejecutando, tomar acciones de mejora continua y proponer planes de acción. Dentro de esta ceremonia utilizamos una metodología llamada DAKI de sus siglas en inglés Drop, Add, Keep, Improve. En el Anexo 11: Reuniones realizadas, mostramos evidencias de las reuniones realizadas.

Gracias a esta pudimos visualizar cuáles eran los puntos a mejorar de forma interna, agregar nuevas actividades en base al feedback del equipo, como también dejando de lado las prácticas no tan apropiadas o actividades que pensamos restaban valor.

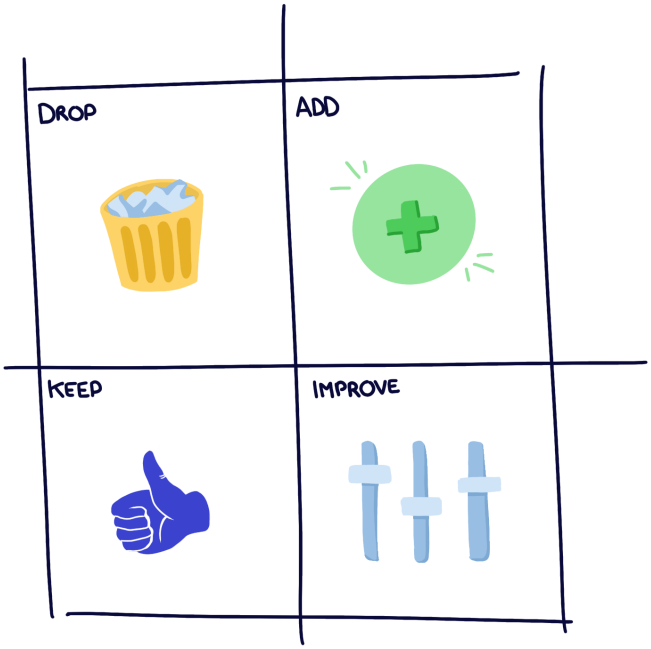


Figura 4.8: Representación de la metodología DAKI.

Daily meeting

Reuniones realizadas a diario entre los integrantes del equipo para evaluar avance del proyecto, definir el foco de cada integrante, además de determinar si existen bloqueos en alguno de los miembros que no le permitan continuar avanzando con el proyecto.

Si bien Scrum propone reuniones diarias, en muchas ocasiones y debido al tamaño del equipo, se trataba de respetar este punto o bien mediante una actualización por algún otro canal de mensajería como forma de dar visibilidad y estado sobre las tareas asignadas.

Para algunas tareas debido a la gran complejidad o poco entendimiento se realizaron actividades de pair-programming, en donde el equipo encontró esta técnica como muy útil cuando se trataba de algún caso similar.

Riesgos

Para cada uno de los riesgos que previmos por cada uno de los releases, elaboramos un plan de respuesta para el caso lleguen a materializarse. Para ver la lista de los riesgos con su plan de respuesta dirigirse al Anexo 3: Planificación de riesgos.

Los riesgos los escribimos con el siguiente formato: “Dado <contexto particular> Cuando <ocurre un evento determinado> Entonces <potencial problema>”.

En la siguiente tabla, mostramos un ejemplo de la forma en la que especificamos cada riesgo.

Riesgo	Plan de respuesta	Descripción
Riesgo 1: No manejar los conocimientos necesarios para desarrollar los módulos de data mining.	Riesgo 1: Capacitar al equipo en las tecnologías necesarias para desarrollar el proyecto, principalmente Machine Learning. Disponer de un referente (technical leader) a quien consultar periódicamente y evitar que este riesgo tenga alto impacto.	Dado el conocimiento que tiene el equipo en las técnicas de data science Cuando se tengan que desarrollar los módulos de identificación de perfiles y sugerencias Entonces es posible que no se tengan los conocimientos suficientes para dichos desarrollos, lo que podría afectar la calidad, la duración y el alcance del proyecto.

Figura 4.9: Tabla con nomenclatura de riesgos.

5. Ejecución del proyecto

En esta sección del documento detallaremos cómo llevamos a cabo el proceso de ejecución del proyecto, desde el relevamiento de requerimientos, diseño, implementación, pruebas y gestión. Todas estas actividades las realizamos basándonos en las metodologías definidas y presentadas en el capítulo anterior.

5.1. Definición de los requerimientos

Como ya dijimos en el capítulo anterior, dada la incertidumbre que manejamos al comenzar el proyecto, la definición de los requerimientos fue pieza clave para la entrega de valor al cliente. En esta sección comentamos cómo se relevaron y especificaron los requerimientos, además de cómo se realizó la definición y manejo del alcance del proyecto. Esto último con el objetivo de lograr completarlo en el tiempo estipulado por la Universidad ORT Uruguay y además entregar las funcionalidades que le aportan el mayor valor al cliente.

5.1.1. Proceso y técnicas de relevamiento

El proceso de relevamiento de requerimientos se desarrolló (como se comentó en el capítulo anterior) en dos etapas. En primer lugar, en la fase de discovery nos reunimos varias veces con el cliente para ir obteniendo conocimiento del negocio y sus necesidades puntuales para este proyecto. A medida que esta fase avanzó fuimos investigando los requerimientos, presentando prototipos livianos (mockups) para ir validando junto con él y además, con ayuda del tutor, ir diseñando una primera versión del product backlog.

Una vez que tuvimos una primera aproximación de los requerimientos, nos reunimos y realizamos un brainstorming que proponía distintas soluciones al problema planteado por el cliente, además de comenzar las reuniones con el experto técnico, quien nos asesoró acerca de la viabilidad de la o las soluciones a considerar.

Cuando finalizamos con la fase de discovery, ya teníamos una primera aproximación de los requerimientos funcionales y no funcionales que iban a formar parte de la solución, además de tener una idea de hacia dónde íbamos a apuntar con la solución.

A medida que avanzaba la fase de delivery y logramos una comprensión más amplia del problema, llegamos a tener una definición más completa de los requerimientos. La especificación de los mismos se terminó de afinar en cada release planning meeting, en la cual dedicamos un gran porcentaje del tiempo a definir las tareas a realizar y las estimaciones correspondientes.

Tanto el Product Backlog completo como los mockups de frontend, se encuentran incluidos en el Anexo 1: Product Backlog y Anexo 2: Mockups de frontend respectivamente.

5.1.2. Requerimientos

Cuando comenzamos a reunirnos con el cliente, éste nos habló de una idea de muy alto nivel de sus necesidades en una charla informal, por lo que la definición y especificación de los requerimientos representó un gran desafío para nosotros.

El cliente nos mencionó que su objetivo principal era esencialmente la identificación de los perfiles de usuario en forma automatizada a partir de la información histórica de compras. Nos manifestó que en el pasado este análisis y agrupamiento se había intentado hacer de manera manual pero por razones de escala de datos no fue posible completarlo.

Como el proyecto se enmarcó dentro de metodologías ágiles (ver más detalle en el capítulo 4. Metodología), los requerimientos fueron especificados en la forma de user stories. Dichas user stories constan de una narrativa, criterios de aceptación, estimación en story points y un conjunto de personas responsables de su desarrollo. La narrativa especificó el rol interesado, la necesidad y el propósito de cada una. En forma genérica especificamos una narrativa de una user story de la siguiente manera: “COMO <rol> QUIERO <necesidad> PARA <propósito>”.

Figura 5.1: Ejemplo de narrativa en la especificación de un requerimiento.

Los criterios de aceptación se especificaron en base a escenarios, donde cada uno define qué respuesta deberá brindar el sistema bajo ciertas condiciones o circunstancias. Como también mencionamos anteriormente, en forma genérica, especificamos los escenarios de la siguiente forma: “GIVEN <condición de entrada> WHEN <cuando ocurre estímulo> THEN <respuesta sistema>”.

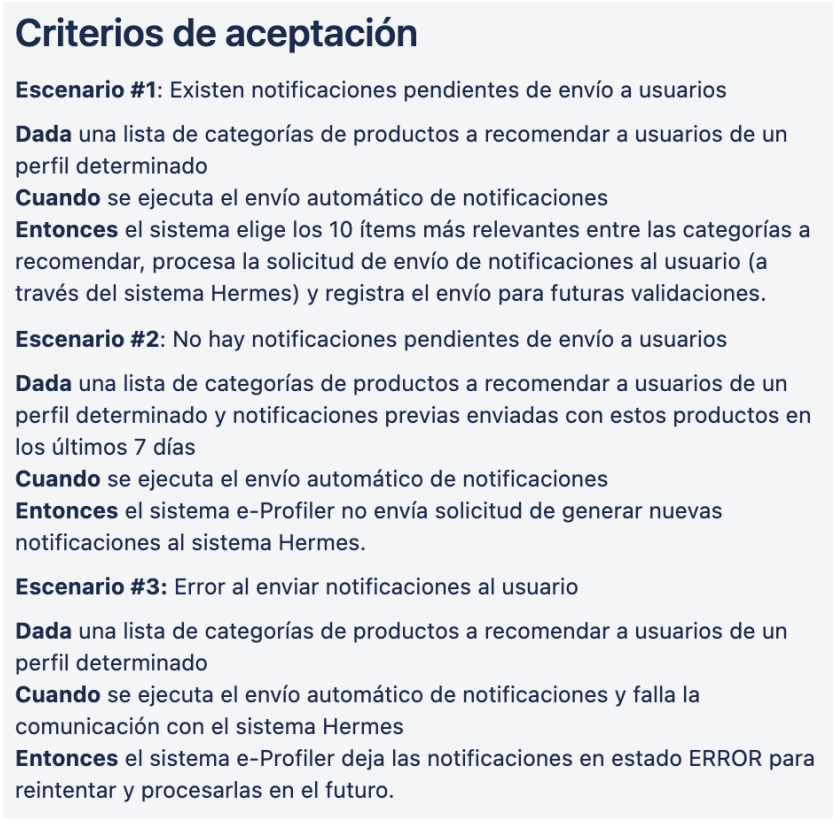


Figura 5.2: Especificación de criterios de aceptación en base a escenarios.

Lo anterior nos resultó beneficioso en el sentido de que nos permite ir detallando los requerimientos del negocio, poder implementarlos como casos de prueba del sistema en lenguaje Gherkin y luego verificar el cumplimiento de cada requerimiento por parte de la implementación.

```
Feature: View Profiles

  View profiles data

  Background: Load page
    Given A page with URL '/profiles'
    And Page structure is loaded

  Scenario: Logged user and empty profiles list
    Given Logged user is defined
    And Profiles list is empty
    When Click on profiles-menu-item menu item
    Then It should show empty profiles data

  Scenario:  Logged user and non-empty profiles list
    Given Logged user is defined
    And Profiles list is not empty
    When Click on profiles-menu-item menu item
    Then It should show non empty profiles data

  Scenario: View users by profile (non-empty profiles list + logged user)
    Given Logged user is defined
    And Profiles list is not empty
    When Click on profile users button
    Then It should show users list by profile option
```

Figura 5.3: Ejemplo de implementación de casos de prueba en lenguaje Gherkin.

La estimación de las historias de usuario se realizó en story points, utilizando la escala de fibonacci y la técnica de affinity estimation, la cual consiste en ir asignando story points en base a la comparación de otras ya estimadas.

Figura 5.4: Ejemplo de una estimación de una user story en trello.

Al utilizar one-piece development, para cada user story definimos piezas paralelizables, según la naturaleza de cada artefacto a desarrollar dentro de ellas (por ejemplo, frontend, backend y data science). Una vez definida cada pieza, se realizaron las estimaciones de esfuerzo de cada una de las tareas en horas, ya que creímos que es la forma más real de entender cuánta dedicación íbamos a necesitar y para comparar con la capacidad del equipo para el release.

5.1.3. Definición de alcance

Debido al amplio universo de usuarios que abarca la empresa Mercado Libre dentro de su área de operaciones en latinoamérica en contraposición al tiempo acotado que disponemos para el proyecto, fue muy importante adaptar el alcance del último para lograr un balance entre las expectativas del cliente y los requisitos académicos.

La gestión de alcance para este proyecto comenzó con las reuniones iniciales con el cliente para recopilar sus necesidades y entender la dimensión real del problema a resolver. En base a las reuniones de relevamiento de requisitos y propuestas de valor presentadas por nosotros para enriquecer el producto final elaboramos el product backlog, el cual se priorizó y se definió el alcance real del proyecto a elaborar en el tiempo establecido por la Universidad ORT Uruguay. Para visualizar el Product Backlog completo dirigirse al Anexo 1: Product Backlog.

Una vez elaborado dicho product backlog, nos centramos en las funcionalidades más prioritarias para nuestro cliente y que a su vez eran las que estimábamos que nos iban a demandar mayor esfuerzo, todo esto con el objetivo de entregar el mayor valor posible al cliente y a su vez lograr completar el proyecto académico en el tiempo establecido.

Considerando el balance mencionado anteriormente, definimos y acotamos el alcance del problema que se pretendía resolver. Luego de esto, ajustamos el product backlog para que cumpla con lo acordado y en cada release planning meeting se realizó la estructura de desglose de trabajo y las estimaciones pertinentes para cada una de las tareas a realizar.

Las tareas de verificación de alcance se llevaron a cabo en las reuniones de release review y retrospective, en las cuales verificamos si realmente estuvimos entregando el producto a la medida acordada con el cliente y obtener su aprobación.

5.2. Desarrollo del proyecto

Una vez que dimos por finalizada la etapa de discovery, en la que definimos el marco metodológico, una primera versión de los requerimientos, las tecnologías y diseñamos la solución, comenzamos con el desarrollo de la solución del proyecto.

Es importante aclarar que durante esta etapa, principalmente cuando se comenzaba cada release, fue necesario profundizar en los requerimientos, para ello fue necesario volver a iterar sobre la fase de discovery principalmente para revisar y ajustar las definiciones respecto al módulo de data mining (identificación de perfiles y recomendaciones). Aquí nos dimos cuenta que no teníamos del todo claros algunos requerimientos y fue necesario volver a investigar más en profundidad.

5.2.1. Seguimiento del proyecto

El seguimiento de las tareas planificadas del proyecto lo fuimos realizando utilizando la funcionalidades de la herramienta Trello. Utilizamos un tablero de Trello para cada release del proyecto, en los que incluimos columnas que corresponden con las distintas fases por las que puede transitar una pieza de software o user story para la metodología correspondiente.

Figura 5.5: Tablero para seguimiento del proyecto (I).

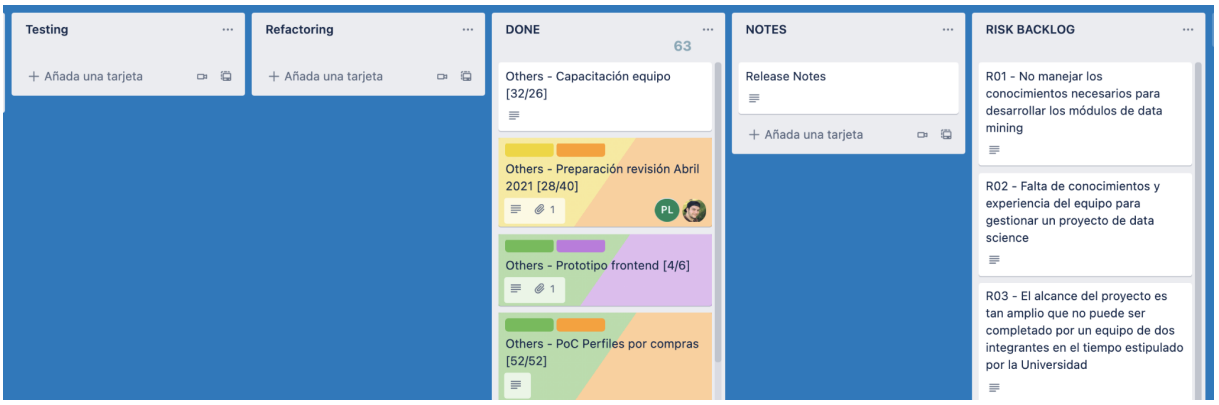


Figura 5.6: Tablero para seguimiento del proyecto (II).

Como describimos en el capítulo 4. Metodología, en cada tarjeta de Trello representamos una pieza de software a desarrollar (por ejemplo frontend, backend o data mining).

En cada tarjeta correspondiente a una pieza de software, además de las estimaciones por fase (correspondiente a cada metodología), fuimos registrando el esfuerzo real dedicado por persona, además del estado actual de la tarea.

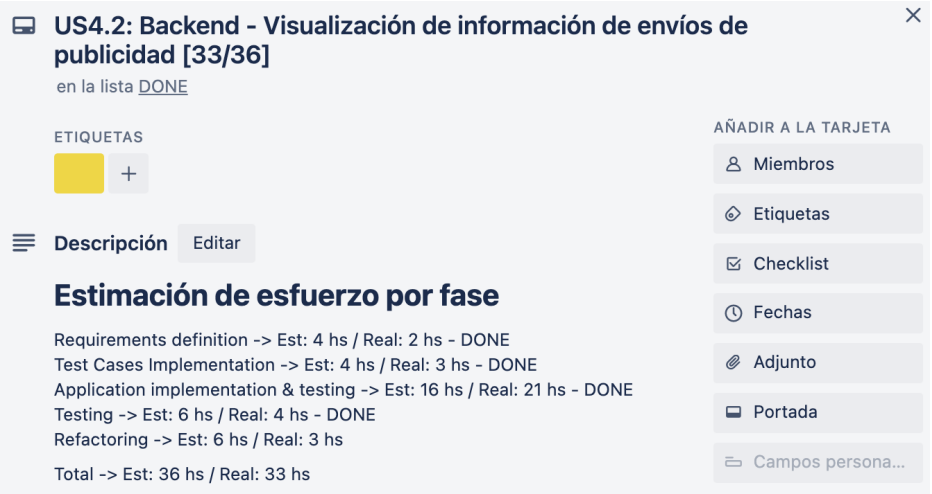


Figura 5.7: Ejemplo de tarjeta de trello con seguimiento por fase de BDD.

Esta información de seguimiento fue recopilada para el relevamiento de información para la métrica de desviación en las estimaciones (por más información referirse al Anexo 6: Definición de métricas). Para esta recopilación utilizamos planillas electrónicas de Google Drive (Google Spreadsheets), las cuales permitieron realizar cálculos y generar gráficos a partir de datos tabulares.

Reuniones de seguimiento y planificación

Como comentamos en la sección 4. Metodología, las ceremonias realizadas dentro del tiempo del proyecto fueron las siguientes:

Release planning meeting: refinamos las stories incluidas en el Product Backlog y confeccionamos el release backlog con las user stories priorizadas para cada release. La distribución de user stories por release se muestran en el Anexo 4: User stories por release.

Release retrospective: reflexionamos sobre el proceso y sobre la ejecución en general del proyecto. En estas reuniones participamos ambos integrantes del equipo de desarrollo y en ocasiones participó nuestro Tutor. En el Anexo 11: Reuniones realizadas se muestran los resultados de las reuniones cumplidas.

Release review: informamos y presentamos el avance del proyecto al cliente, mostrando la versión funcionando en el ambiente productivo, con el objetivo de mostrar el trabajo realizado y recibir su feedback. En el Anexo 11: Reuniones realizadas adjuntamos minutas de las reuniones realizadas con el cliente.

Revisión de plan de riesgos: al finalizar cada release en la reunión de retrospective realizamos la revisión del plan de riesgos. En el Anexo 3: Planificación de riesgos mostramos la planificación y revisión de los riesgos por release.

5.3. Cronograma

Para comenzar presentamos el esquema del cronograma que planificamos inicialmente para el proyecto. Como comentamos anteriormente el tiempo del proyecto se dividió en dos fases, una de discovery y otra de delivery.

Inicialmente estimamos dedicar tres meses del proyecto, comenzando en octubre de 2020 y finalizando en enero de 2021. A continuación presentamos un esquema que muestra las fechas mencionadas además de la fecha de finalización del proyecto, establecida por la Universidad ORT Uruguay para el 30 de setiembre de 2021.

Figura 5.8: Planificación inicial de cronograma.

Planificación de releases y secuenciamiento de tareas

A continuación presentamos la planificación de los releases pactados con el cliente, las fechas y features a incluir en cada uno de ellos. La idea fue ir entregando resultados incrementales al cliente cada determinado número de meses.

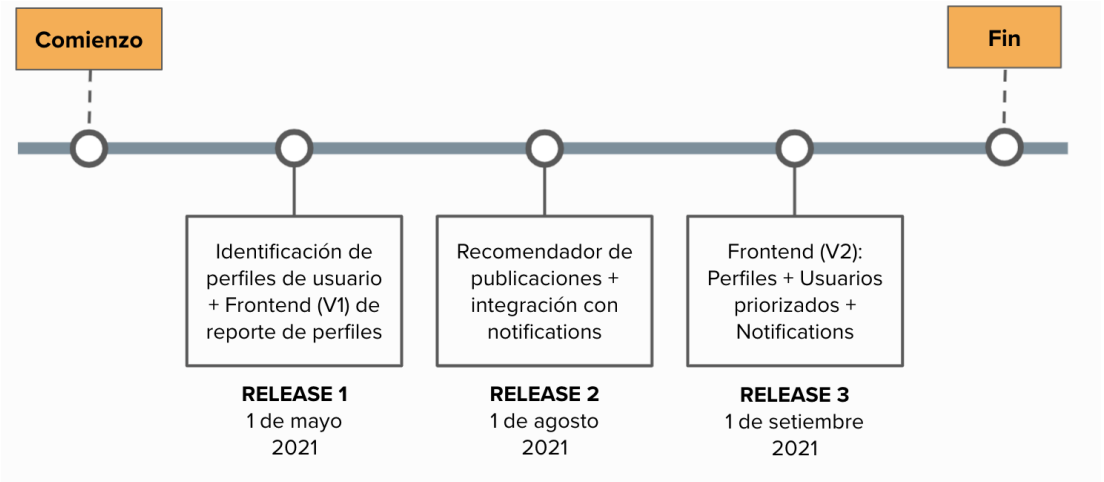


Figura 5.9: Planificación de user stories por release.

Control y seguimiento del cronograma

Las actividades relacionadas para este punto se llevaron a cabo en la release planning meeting, retrospective, reuniones con el tutor y cliente. En estas instancias evaluamos el avance del proyecto, los riesgos que se fueron materializando, convirtiéndo en problema y en qué medida fue necesario re-planificar el cronograma planificado inicialmente.

Variación del cronograma en relación a lo planificado

Debido a que nos vimos obligados a extender la fase de discovery, como comentamos en la sección de metodología, tuvimos que aplazar un release para una etapa post proyecto. Esta decisión fue negociada con el cliente y fue aceptada por el mismo, bajo la condición de mantener los requisitos que él mismo priorizó.

A continuación presentamos cómo se modificó el cronograma y se mantuvieron las prioridades del cliente, además de las fechas de entrega pactadas.

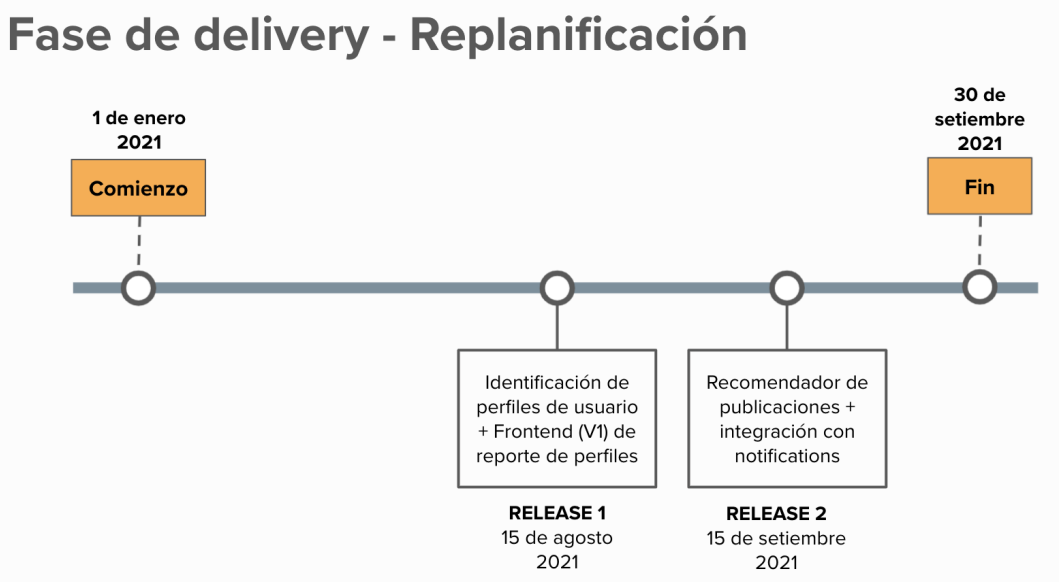


Figura 5.10: Planificación de user stories por release.

Cabe aclarar que las funcionalidades que integraban el tercer release 3 (Frontend V2 (Perfiles + Usuarios priorizados + Notifications), fueron pospuestas en común acuerdo con el cliente para una etapa posterior a la entrega del proyecto académico. Las funcionalidades que se pospusieron las incluimos dentro de la sección 8. Next Steps, junto con otros next steps del proyecto en general.

En las secciones de gestión de riesgos y seguimiento del cronograma del proyecto presentamos en más detalle cómo fue evolucionando el cronograma y en qué medida se vio afectado por las situaciones que fueron surgiendo.

5.4. Métricas

En esta sección presentamos una breve descripción de las métricas que definimos para el proyecto, las cuales agrupamos en métricas de negocio, gestión y calidad de la solución. Para cada métrica definida presentamos el valor obtenido y su comparación con el valor meta para este proyecto. La definición completa de las métricas (motivación, objetivos a los que contribuye, forma de obtención de los datos y valores objetivo) se encuentra disponible en el Anexo 6: Definición de métricas.

5.4.1. Métricas de negocio

Frecuencia de compra de usuarios

Representa la métrica más importante para el cliente y el negocio. El valor relevado para esta métrica para el año 2020 fue de 31,8 días entre compras, mientras que el valor meta de este proyecto fue estimado en 20 días entre compras (para más detalles dirigirse al Anexo 6: Definición de métricas).

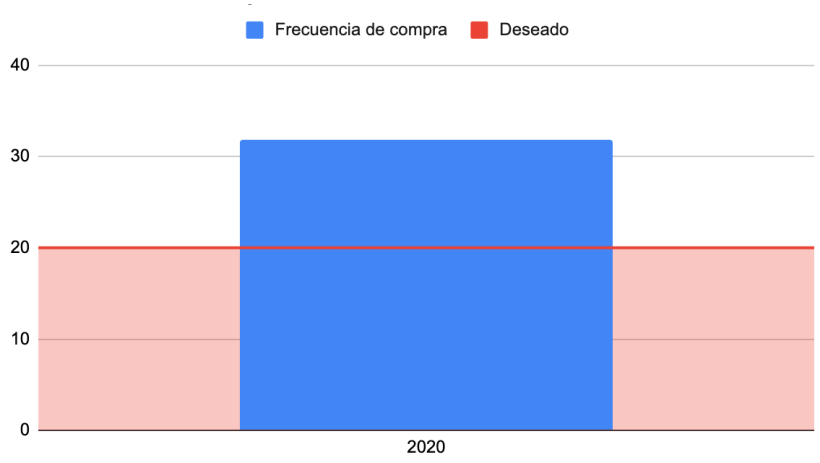


Figura 5.11: Gráfico de valor de frecuencia de compras para el año 2020.

Tiempo de ejecución del proceso de ETL

Refiere al tiempo que insume la ejecución del proceso de extracción, transformación y carga de datos para el proceso de identificación de perfiles y recomendaciones.

Esta métrica fue obtenida a través de los resultados obtenidos por parte de la suite Fury una vez realizadas las distintas ejecuciones.

A continuación mostramos la siguiente gráfica con los valores reales obtenidos como también los máximos preestablecidos en base a la ejecución de procesos de similares características.

Cabe destacar que tanto los tiempos máximos como insumidos son acorde a la cantidad de datos. Por esta razón generamos el siguiente cuadro comparativo.

	Tiempo insumido (min)	Máximo deseado (min)
30 días de compras	92	240
60 días de compras	112	240
90 días de compras	168	240

Figura 5.12: Tabla con tiempos de ejecución de procesos de ETL según ventana de tiempo.

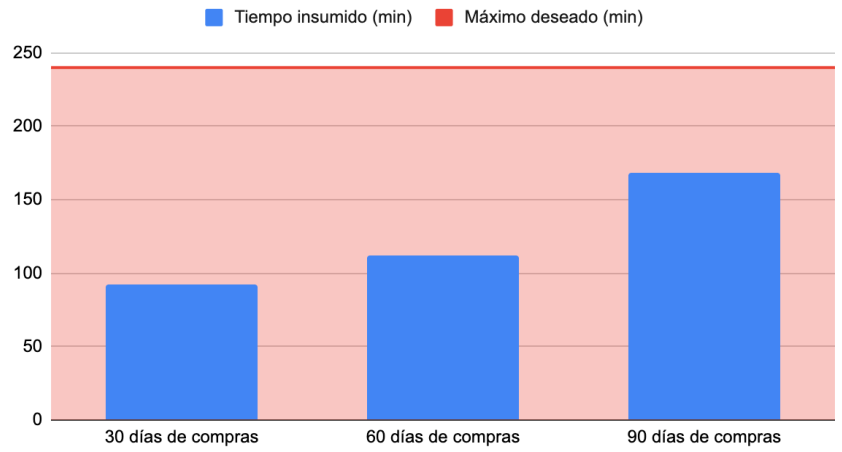


Figura 5.13: Gráfico con tiempos de ejecución del proceso de ETL.

Grado de satisfacción del cliente con el producto

Refiere a la medida en que el cliente se encuentra satisfecho con el producto entregado.

	Satisfacción	Mínimo deseado
Release 1	9	8
Release 2	9	8

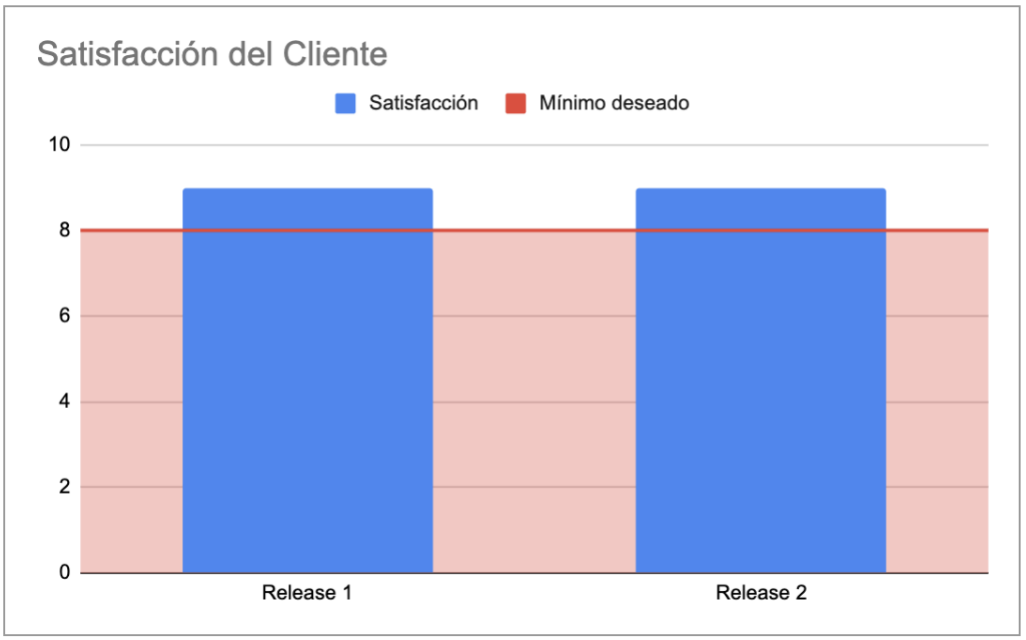


Figura 5.14: Gráfico con niveles de satisfacción del cliente por release.

5.4.2. Métricas de gestión

Desviación en estimaciones y distribución del esfuerzo

Refiere a la diferencia entre el esfuerzo real y el estimado (en horas).

	Desviación	Mínimo deseado
Release 1	36%	20%
Release 2	11%	20%

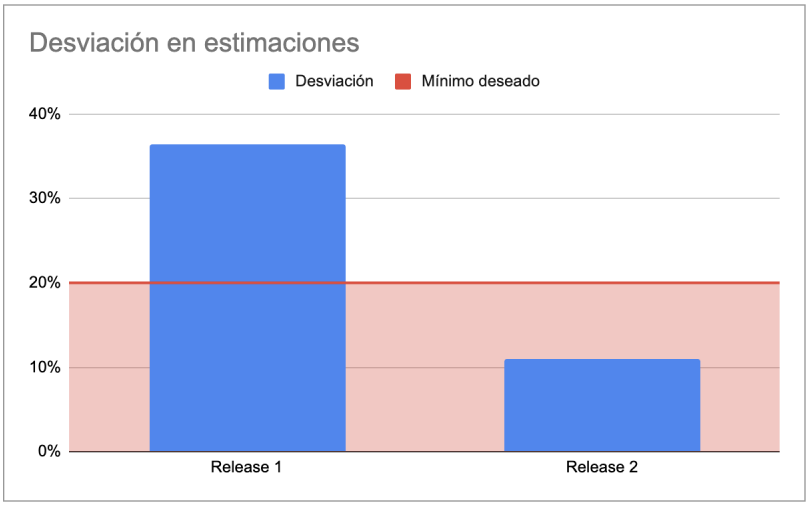


Figura 5.15: Desviación en estimaciones por release.

De este gráfico podemos mencionar varios comentarios y conclusiones. En primer lugar podemos decir que en el release 1 estuvimos muy por encima (casi el doble) del máximo porcentaje de desviación establecido para el proyecto, lo que nos habla de que la incertidumbre en los requerimientos no había sido reducida a un nivel aceptable y los conocimientos adquiridos en materia de proyectos de data science no habían sido del todo suficientes para el desarrollo del producto. A medida que avanzó el tiempo y pasamos al release 2, logramos reducir esa incertidumbre y, con más experiencia del release anterior (como consecuencia del aprendizaje contínuo adquirido durante el proceso y reuniones de retrospectiva), pudimos corregir las estimaciones, lo que redujo el porcentaje de desviación y nos permitió finalizar el release 2 con un valor de la métrica por debajo del valor deseado.

Los gráficos con las distribuciones de esfuerzo por release los presentamos en el Anexo 5: Seguimiento de los releases del proyecto.

5.4.3. Métricas de calidad de la solución

Usabilidad del frontend

Refiere a la métrica de calidad del frontend de la solución, basado en heurísticas de Nielsen.

Heurística	Valor	Umbral mínimo
Visibilidad del estado del sistema	4	3,5
Coincidencia entre el sistema y el mundo real	4,2	3,5
Control y libertad del usuario	3,6	3,5
Prevención de errores	4,4	3,5
Mostrar en lugar de recordar	3,8	3,5
Flexibilidad y eficiencia de uso	4,4	3,5
Diseño estético y minimalista	4,8	3,5
Ayudar a reconocer, diagnosticar y recuperarse de errores	4,2	3,5
Ayuda y documentación	1,8	3,5
Promedio	3,91	3,5

Figura 5.16: Tabla con heurísticas de Nielsen y los valores relevados.

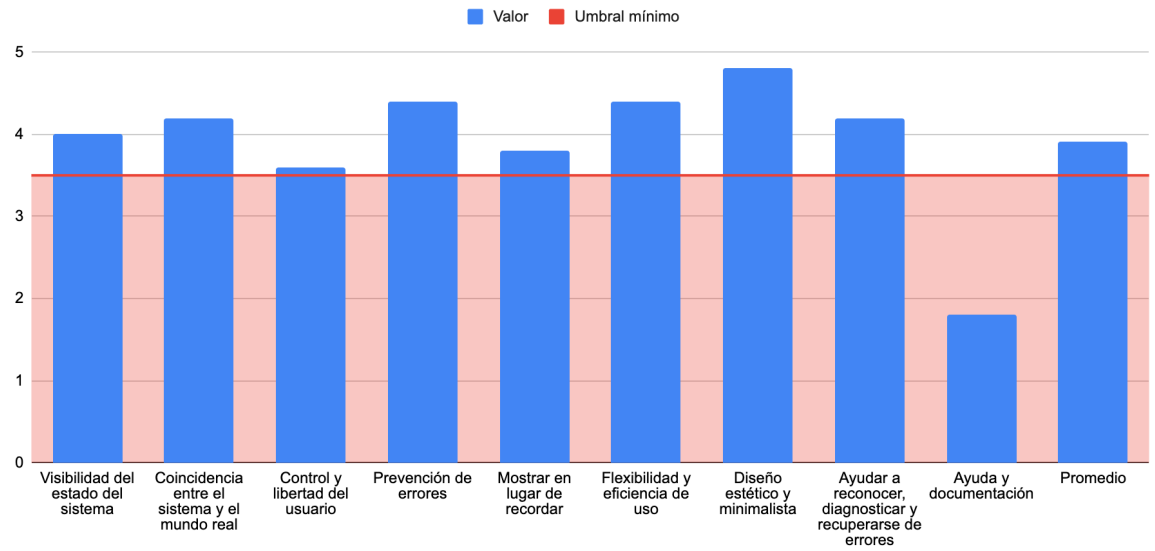


Figura 5.17: Gráfico con valores de las heurísticas de Nielsen.

Haciendo una lectura acerca de los valores recogidos en relación a los distintos puntos que componen la usabilidad del frontend, podemos decir que en 8 de los 9 puntos relevados estamos por encima del mínimo establecido por el equipo como aceptable. El único punto que no cumplimos en llegar al mínimo propuesto fue en “Ayuda y Documentación”, el cual creemos que es una oportunidad interesante de

mejora del producto. Por razones de prioridades y tiempos no incluimos un manual de usuario, que permita al operador de Mercado Libre tener una referencia rápida del uso de la herramienta y no requiera solamente de su intuición para utilizarla. De todas formas creemos que el hecho de cumplir con 8 de 9 puntos nos da la pauta de que la herramienta cumple en forma satisfactoria con los principios de usabilidad más importantes y tiene oportunidades de mejora para el futuro.

Porcentaje general de cobertura de código

Refiere al porcentaje de código cubierto con pruebas automatizadas, por release.

	Test Coverage	Umbral mínimo
Release 1	88%	80%
Release 2	88%	80%

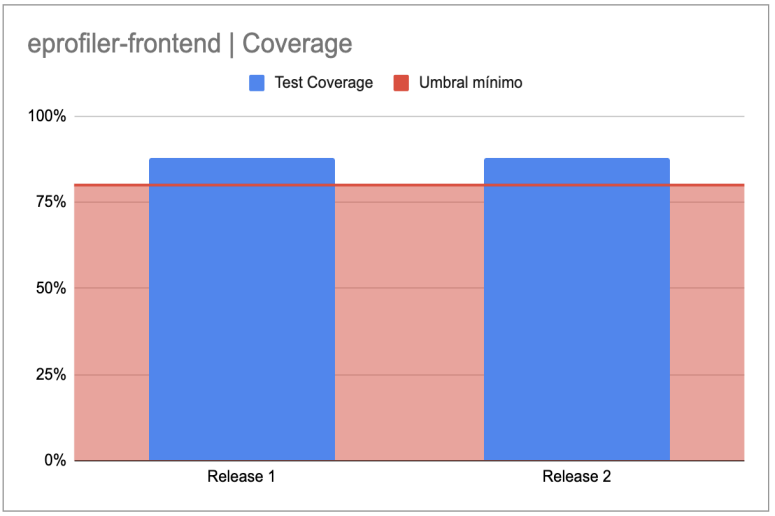


Figura 5.18: Gráfico con valores de coverage por release para la aplicación de frontend.

	Test Coverage	Umbral mínimo
Release 1	88,76%	80%
Release 2	93,30%	80%

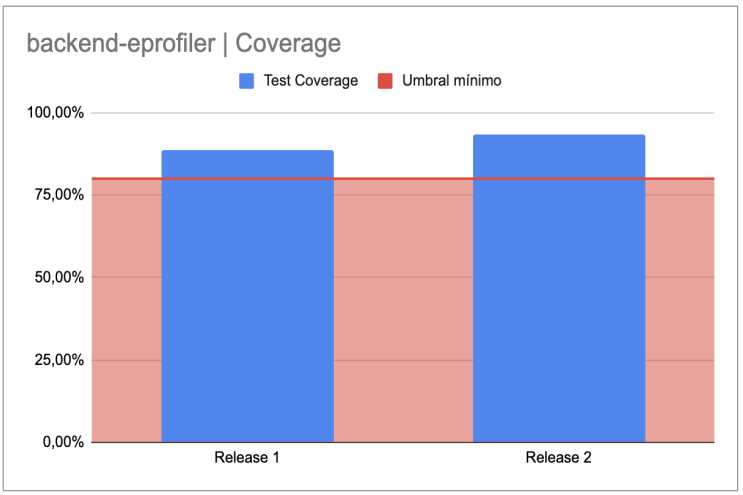


Figura 5.19: Gráfico con valores de coverage por release para la aplicación de backend.

5.5. Aseguramiento de la calidad

En esta sección del documento comentamos cómo realizamos el aseguramiento de calidad en los distintos aspectos que refieren al proyecto.

La calidad en el contexto de este proyecto la definimos en función de tres parámetros fundamentales, en primer lugar la especificación de requerimientos, la solución y la documentación. El punto que generó mayor incertidumbre y requirió más esfuerzo de investigación fue el chequeo de calidad de los módulos de inteligencia artificial, por tratarse de una aplicación de distinta naturaleza a las tradicionales que se desarrollan a diario en el ambiente laboral o educativo.

5.5.1. Calidad en la especificación de los requerimientos

En la especificación de los requerimientos definimos escenarios relevados en base a las necesidades del cliente e incluimos casos de prueba dentro de la solución para verificar que ésta cumple con las especificaciones establecidas por el cliente.

Para la validación de la especificación de los requerimientos con el cliente en etapas tempranas utilizamos la técnica de prototipación, en las que presentamos bosquejos gráficos del frontend (mockups) y recibimos su feedback. En el Anexo 2: Mockups de frontend presentamos los mockups presentados al cliente.

5.5.2. Calidad de la solución

Para asegurar la calidad de la solución y que ésta cumple con los requerimientos relevados de las reuniones con el cliente, realizamos cuatro tipos de pruebas: unitarias, de integración, funcionales y pruebas manuales en ambiente productivo. A continuación detallamos las técnicas y herramientas utilizadas.

Pruebas unitarias

Para este tipo de prueba, utilizamos el framework JUnit provisto dentro del stack de Mercado Libre para aplicaciones Java. El objetivo de este tipo de prueba fue verificar el comportamiento de los métodos aislados del resto en relación a la especificación de cada uno.

Pruebas de integración

En este tipo de prueba, para la aplicación del backend también utilizamos el framework JUnit con la extensión de Mockito, y tuvo el cometido de probar el comportamiento de los distintos módulos de la aplicación en interacción con otros, utilizando mocks con el objetivo de emular todos los flujos posibles que pueden darse dentro del funcionamiento de la aplicación.

```java
public class ProfileServiceTest extends EProfilerServiceTest {

    @Mock
    private SystemUserService systemUserServiceMock;

    @Mock
    private ProfilesRepository profilesRepositoryMock;

    @InjectMocks
    private ProfileServiceImpl profileService;

    @Before
    public void setup() { MockitoAnnotations.openMocks( testClass: this); }

    @Test
    public void getAllProfilesWhenUserIsNotFound() {
        //Given
        Mockito.when(systemUserServiceMock.getUserByNickname("user")).thenReturn(null);

        //When, Then
        assertThrows(NotFoundElementException.class, () -> profileService.getAllProfiles( userId: "user"));
    }

    @Test
    public void getAllProfilesWhenUserIsFoundButIsNotManager() {
        //Given
        Mockito.when(systemUserServiceMock.getUserByNickname("user")).thenReturn(getSampleMarketingOperator());

        //When, Then
        assertThrows(ForbiddenException.class, () -> profileService.getAllProfiles( userId: "user"));
    }

    @Test
    public void getAllProfilesUserIsAllowed() throws EProfilerException {
        //Given
        Mockito.when(systemUserServiceMock.getUserByNickname("user")).thenReturn(getSampleSystemManager());
        Mockito.when(profilesRepositoryMock.findAll()).thenReturn(getSampleProfilesList());

        //When
        List<Profile> profileList = profileService.getAllProfiles( userId: "user");

        //Then
        assertEquals( expected: 2, profileList.size());
    }
```

Figura 5.20: Ejemplo de Implementación de test de integración.

Estas pruebas, al igual que las unitarias, fueron ejecutadas en forma automatizada al momento de integrar el código a la rama principal de trabajo.

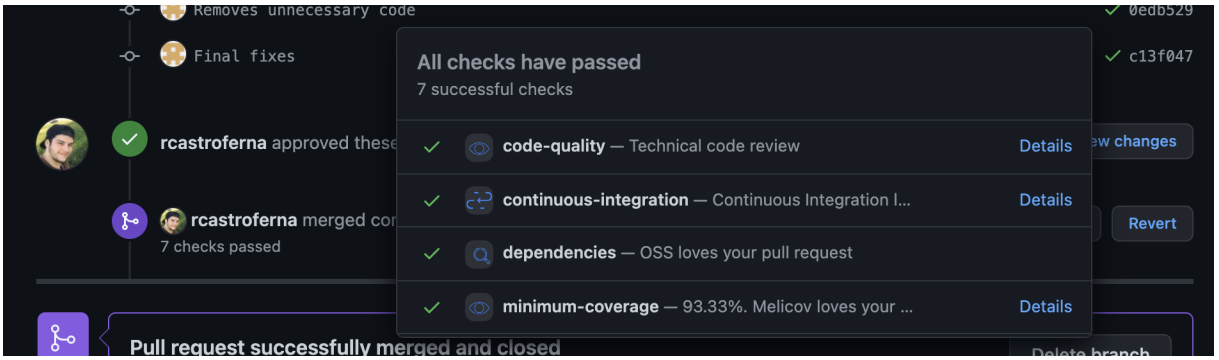


Figura 5.21: Ejecución de los chequeos al integrar el código a la rama principal.

Pruebas funcionales basadas en escenarios

Dentro del marco de trabajo BDD, implementamos tests funcionales basados en escenarios para el backend y el frontend de la aplicación de e-Profiler.

Los escenarios fueron implementados en el lenguaje Gherkin y las pruebas sobre ellos fueron ejecutadas con el framework Cucumber (utilizando la SDK de Cucumber para Java o Node JS respectivamente).

Al ejecutar estos tests basados en escenarios, Node JS ejecuta los tests automáticamente en el navegador utilizando la herramienta Selenium.

A continuación presentamos un ejemplo de implementación de escenario que se ejecuta automáticamente y el resultado de la ejecución de dicha prueba.

```
ChromeDriver was started successfully.
[0-2] [WDIO] SELENOID HOST local.adminml.com
[0-0] [WDIO] SELENOID HOST local.adminml.com
[0-1] [WDIO] SELENOID HOST local.adminml.com
[0-1] RUNNING in chrome - /tests/e2e/features/Roles.feature
[0-2] RUNNING in chrome - /tests/e2e/features/ViewProfiles.feature
[0-0] RUNNING in chrome - /tests/e2e/features/Notifications.feature
[1632540076.544][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
[1632540076.572][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
[1632540078.024][WARNING]: This version of ChromeDriver has not been tested with Chrome version 93.
[0-0] PASSED in chrome - /tests/e2e/features/Notifications.feature
[0-2] PASSED in chrome - /tests/e2e/features/ViewProfiles.feature
[0-1] PASSED in chrome - /tests/e2e/features/Roles.feature
```

Figura 5.22: Resultado de ejecución de test basados en escenarios con cucumber.

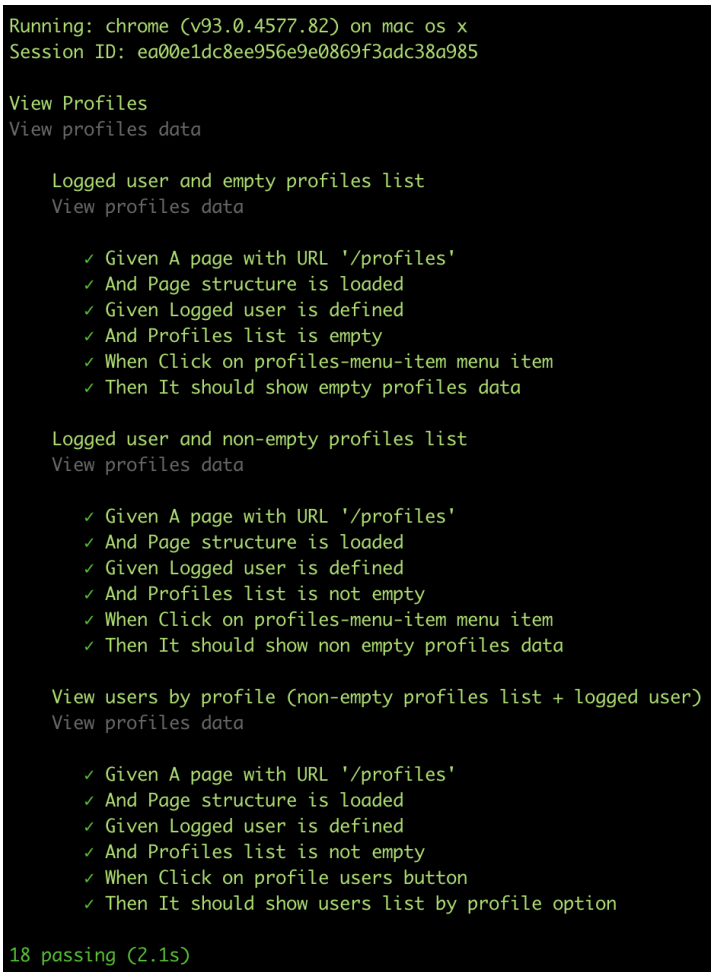


Figura 5.23: Resultado de ejecución de tests implementados con escenarios.

5.5.3. Calidad en los módulos de inteligencia artificial

Este punto creemos que fue el más desafiante para las tareas de aseguramiento de calidad de los módulos de inteligencia artificial, que como ya comentamos, por la naturaleza del proyecto, la incertidumbre que manejamos como equipo.

Las tareas de aseguramiento de calidad las dividimos en dos partes fundamentales, por un lado una validación manual en base a un muestreo de datos y por otro lado la calidad en términos de métricas de negocio.

5.5.4. Otros chequeos de calidad

Cobertura de código

Otro punto importante en el aseguramiento de la calidad de la solución (también relacionado con las métricas de éxito del proyecto) fue la cobertura de código. Este

chequeo de la herramienta Jacoco se ejecutó automáticamente al momento de integrar el código de una feature branch en la rama principal.

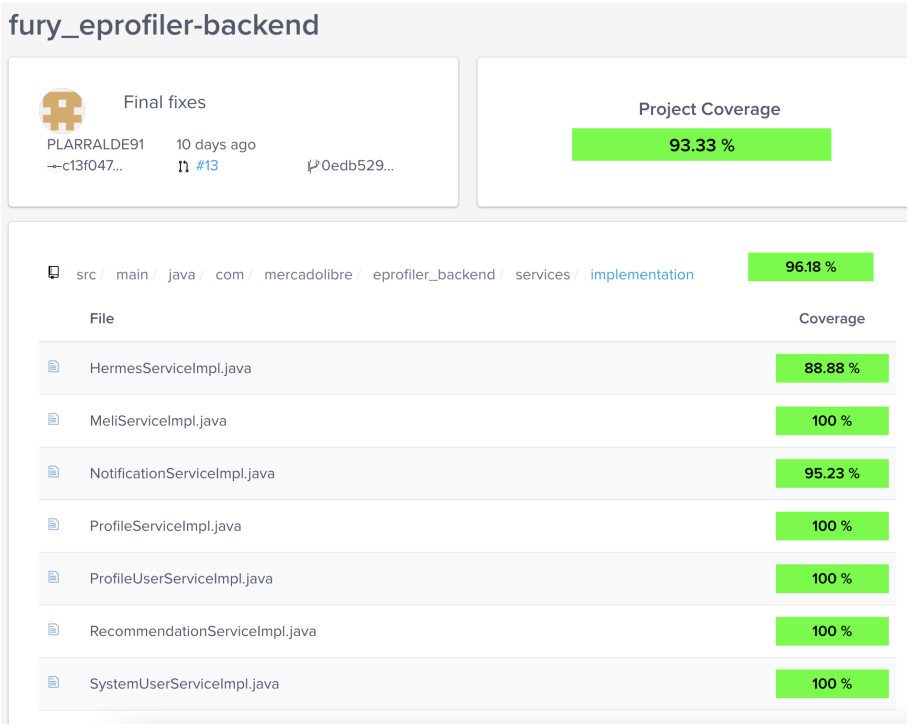


Figura 5.24: Ejemplo de reporte coverage resultante de la herramienta Jacoco.

Buenas prácticas de codificación

Dentro del workflow de la aplicación de frontend, al integrar el código a la rama principal se ejecutó un chequeo sintáctico y de buenas prácticas de codificación. Este chequeo tiene como objetivo mantener un estándar de codificación de todas las aplicaciones desarrolladas dentro de Mercado Libre.

Mostramos un ejemplo donde se visualizan errores y advertencias a corregir para que se permita la integración de código. Si los errores no son corregidos, fallarán los tests y no podrá integrarse el código a la rama principal.

```
10:33  warning  `frontend-restclient/src/build-flow-context` import should occur before import of `./views/Profiles`  import/order
42:2   error    Missing semicolon                                                                                     semi
51:11  error    There should be no space after this paren                                                             space-in-parens
56:16  error    Missing semicolon                                                                                     semi
61:7   error    'role' is never reassigned. Use 'const' instead                                                       prefer-const
62:3   error    Expected space(s) after "if"                                                                          keyword-spacing
62:11  error    Expected '!==' and instead saw '!='                                                                   eqeqeq
63:34  error    Missing semicolon                                                                                     semi
67:2   error    Missing semicolon                                                                                     semi
75:45  error    Trailing spaces not allowed                                                                           no-trailing-spaces
76:11  error    There should be no space after this paren                                                             space-in-parens
81:16  error    Missing semicolon                                                                                     semi
83:3   error    Newline required at end of file but not found                                                         eol-last
```

Figura 5.25: Visualización de errores en el linters.

5.5.5. Calidad en la documentación

La calidad del presente documento la aseguramos apegándonos al estándar definido por la Universidad ORT Uruguay para este tipo de proyectos de Ingeniería en Sistemas junto con los lineamientos y revisiones del tutor del proyecto.

5.5.6. Conclusiones

Para finalizar la sección de aseguramiento de la calidad, presentamos conclusiones acerca de las pruebas realizadas y los resultados obtenidos.

En cuanto a los requerimientos y el cumplimiento de los criterios de aceptación (especificados en base a escenarios) podemos decir que nuestras aplicaciones de frontend y backend cumplen con todos los requisitos definidos en conjunto con el cliente, además de cumplir con los chequeos de cobertura de testing automatizado requeridos para la integración del código a producción.

Si nos referimos a los módulos de data mining (identificación de perfiles y recomendaciones) podemos decir que (por la naturaleza e incertidumbre de éstos), si bien realizamos muestreos manuales internamente en el equipo, con el cliente y expertos técnicos de Mercado Libre, los cuales resultaron satisfactorios desde el punto de vista de negocio, quien dará la pauta real de la calidad del producto será la métrica de frecuencia de compra. Esta métrica definirá si realmente el producto es de calidad (desde el punto de vista de utilidad) para el cliente, más allá de cumplir con la especificación de los requerimientos funcionales y no funcionales.

En general creemos que nuestra solución cumple ampliamente con los estándares de calidad definidos por Mercado Libre para sus aplicaciones, desde el punto de vista de las validaciones y chequeos necesarios para la integración de un código a producción, pruebas basadas en escenarios para garantizar el cumplimiento de las especificaciones de los requerimientos y una primera revisión manual entre varios actores del proyecto de los módulos de data mining, la cual arrojó resultados coherentes y satisfactorios desde el punto de vista de negocio.

5.6. Gestión de la configuración

En este apartado presentamos cómo realizamos las operaciones de gestión de la configuración. Mostraremos cómo realizamos el manejo del control de las versiones de código fuente y documentación, además de los procesos de integración contínua, control de cambios y defectos, todo dentro del marco del stack tecnológico y de herramientas provistas por Mercado Libre.

5.6.1. Control de versionado

Código fuente

El flujo de trabajo definido por la compañía para este punto está basado en Git Flow, el cual establece una serie de normas que deben cumplir los branches y las versiones generadas para que sean aceptadas. Este flujo de trabajo lo detallamos en el Anexo 10: Fury Workflow (Gitflow).

La generación de las distintas versiones de las aplicaciones de frontend, backend y data mining son generadas a través de la herramienta de generación de versiones de Fury, la cual también resuelve las integraciones con github para etiquetado y empaquetado de dichas versiones.

Documentación

El versionado de la documentación lo realizamos utilizando la herramienta online Google Drive, la cual nos fue manteniendo las distintas versiones de la misma. El uso de una herramienta colaborativa online nos permitió trabajar en paralelo y en forma coordinada entre el equipo y el tutor del proyecto, quien nos fue realizando sugerencias y comentarios que permitieron enriquecer y evolucionar cada parte hasta el armado del documento final.

5.6.2. Integración contínua

Para este proyecto utilizamos el sistema de integración contínua provisto por la herramienta Fury (estándar de la empresa), la cual provee un mecanismo automatizado (Jenkins) cuya responsabilidad es ejecutar los chequeos definidos

para cada tipo de aplicación, además de generar y disponibilizar las versiones que se van a poner en producción.

A continuación presentamos un ejemplo de flujo de ejecución de Jenkins al crear una versión de la aplicación del frontend.

Figura 5.26: Ejemplo de workflow de integración contínua de aplicación de frontend.

Chequeos de integración

Dentro del sistema de integración contínua, al momento de integrar una nueva feature al branch principal de desarrollo, el sistema automatizado de Fury (integrado con github) ejecuta distintos chequeos que pretenden asegurar que el código a integrar es compatible con los estándares definidos además de no perjudicar al ya existente.

Alguno de los chequeos que se ejecutan son los siguientes:

- **Testing:** se ejecutan los tests unitarios, de integración y funcionales (si existen).
- **Linters:** se verifica que el código cumpla con los estándares de sintaxis requeridos para la integración.
- **Coverage:** se verifica que la cobertura de test del código del repositorio en general y del código a integrar.
- **Vulnerabilidades:** se verifica que no existan vulnerabilidades dentro del código a integrar. Esto puede darse debido a credenciales incluidas dentro del código o potenciales problemas de seguridad que puedan surgir luego de integrado el código.

Detección de bugs, monitoreo y rollbacks

En caso de ocurrir algún inconveniente con el código integrado de la solución, Fury nos provee un framework de alertas conectado con otras herramientas (como por ejemplo OpsGenie), que nos permite emitir alertas y enviarlas a la persona que se encuentre de guardia en ese momento, de esta forma será notificado del eventual inconveniente y monitorearlo, además de tener la posibilidad de accionar para resolverlo.

De la mano de lo anterior, también disponemos de herramientas de monitoreo de logs, por ejemplo Kibana, que nos permite visualizar los logs de la aplicación en tiempo real y en forma centralizada. Esta herramienta es utilizada en forma estándar por la compañía y nos permite establecer filtros sobre dichos logs para detectar posibles inconvenientes con las versiones que estén en producción o en test.

Puestas en producción de las distintas versiones

Dentro de Fury existen herramientas ya estandarizadas y utilizadas por la compañía para la puesta en producción de las versiones de las distintas aplicaciones (en nuestro caso para frontend, backend y aplicaciones de data mining).

La herramienta nos brindó distintas estrategias para realizar las puestas en producción de las distintas versiones. Por un lado, existen estrategias que están diseñadas para minimizar el downtime en aplicaciones que tengan impacto directo en el usuario final (por ejemplo blue green) y por otro lado existen otras estrategias (como por ejemplo all in) que las utilizamos para disponibilizar versiones en scopes de test o en scopes no interactivos con el usuario (por ejemplo procesos batch).

La aplicación de identificación de perfiles y recomendaciones, como comentamos en la presentación de la solución, consta de tres partes fundamentales: el módulo de ETL, el módulo de train y la interfaz MPI.

Las versiones de estos módulos se llevan a producción de manera independiente y en distintos scopes, lo que garantiza el aislamiento entre ellas ante eventuales fallas y nos permite volver a versiones anteriores en caso de que ocurran.

Cabe agregar que cada versión tiene una serie de metadatos asociados, como su identificador, status, usuario generador, fecha en que se generó, además de link al tag del repositorio de github que almacena dicha versión.

5.6.3. Control de cambios

Dada la naturaleza de la compañía, acompañando los constantes cambios que afronta el negocio del e-commerce actualmente, creímos que este punto iba a ser de mucha utilidad e importancia para nuestro proyecto.

Como tenemos un tiempo limitado para la ejecución del proyecto académico y es alta la probabilidad de que surjan cambios sobre la especificación del producto especificado inicialmente, definimos un procedimiento para gestionarlos e integrarlos en la medida que sea posible.

Establecimos que cada cambio requerido por el cliente o que surja como iniciativa del equipo de desarrollo será ubicado en la siguiente matriz según el esfuerzo requerido y la urgencia y la acción a tomar será la que corresponda según el cuadrante.

Esfuerzo \ Urgencia	Baja	Media	Alta
Bajo	El cambio se toma luego del proyecto académico como next-step	Se toma el cambio dentro del release	Se toma el cambio dentro del release
Medio	El cambio se toma luego del proyecto académico como next-step	Se evalúa el impacto en negocio y proyecto del cambio y se re-prioriza el release	Se evalúa el impacto en negocio y proyecto del cambio y se re-prioriza el release
Alto	El cambio se toma luego del proyecto académico como next-step	Se evalúa el impacto en negocio y proyecto del cambio y se re-prioriza el release	Se evalúa el impacto en negocio y proyecto del cambio y se re-prioriza el release

Figura 5.27: Estrategias de control de cambios según esfuerzo y urgencia.

6. Lecciones aprendidas

En esta sección enumeramos las lecciones aprendidas durante el transcurso del proyecto. Todas ellas surgen de experiencias que vivimos y están formuladas como recomendaciones que hacemos como equipo a otras personas que comienzan un proyecto del mismo estilo.

1) Si se pretende evitar retrabajos y pérdidas de tiempo, entonces se deben establecer claramente las responsabilidades y no superponerlas entre los integrantes del equipo.

En varias ocasiones, sobre todo al comienzo del proyecto, nos sentimos un poco desordenados en cuanto a las responsabilidades que cada integrante tenía asignadas. Por lo anterior decidimos hacer una división de responsabilidades más estricta en la cual nos repartimos las tareas de forma que cada uno de nosotros tuviera una única responsabilidad y no se enfoque en varios temas a la vez.

2) Si tanto el cliente como el Technical Expert no disponen de tiempo para reunirse más de 15 minutos a definir/validar los aspectos del proyecto, entonces debemos anticiparnos y enviar previamente dichas definiciones/validaciones vía mail, de modo que la reunión sea solamente una sincronización y profundización sobre los temas que no hayan podido acordarse.

Este punto fue de vital importancia para la comunicación con el cliente, ya que por sus responsabilidades no disponen de tiempo suficiente para dedicar a una reunión con nuestro equipo para muestras de avance, validación de requerimientos, etc. Esta estrategia (extraída de las políticas de reuniones efectivas de la empresa) nos fue de gran utilidad para no perder continuidad en la comunicación y continuar el trabajo codo a codo con el cliente.

3) Si se trata de un proyecto mayoritariamente orientado a data science, entonces debe utilizarse una combinación de metodologías ágiles con otros marcos de trabajo más específicos para proyectos de este tipo.

Durante el proyecto, y como mencionamos en la sección de Metodologías, hemos descubierto que existen metodologías más apropiadas para los proyectos de data science y también hemos descubierto que las actividades que proponen los enfoques ágiles no encajan tan bien para los proyectos de data science, donde el proceso que comienza con el análisis de datos hasta la evaluación requiere un enfoque distinto y actividades particulares.

4) Si el proyecto requiere pre-procesamiento de datos para obtener resultados intermedios, entonces debe considerarse como un punto crítico de falla y de demora del proceso general.

Este es un punto clave que puede determinar que un proyecto de data science sea exitoso o no, dado que creemos que es fundamental la compatibilidad de los requerimientos del cliente en cuanto a disponibilidad de la información y la capacidad de los procesos de extracción, transformación y carga de datos (ETL). En nuestro caso nos ocurrió que tuvimos algunos inconvenientes con el manejo de memoria para la preparación de datos para los algoritmos de identificación de perfiles y recomendaciones, por lo que pensamos que es un punto crítico a tener en cuenta. Gracias a la ayuda de expertos de la compañía pudimos sortear estos obstáculos y seguir adelante.

5) Si se desea que el sistema cumpla lo más posible los atributos de calidad requeridos, entonces se debe dedicar el mismo tiempo al diseño arquitectónico que a la implementación de las features. Es preferible retrasar un poco el desarrollo y hacerlo sobre una estructura firme.

Como nos han enseñado en toda la carrera, es importante tener una idea lo más afinada posible del diseño y la arquitectura de la solución antes de comenzar con cualquier tipo de implementación o puesta en producción. El hecho de determinar cómo se reflejan los requerimientos no funcionales en la arquitectura es un punto clave para elaborar una solución de calidad.

En este proyecto para algunas ocasiones, si bien diseñamos una arquitectura inicial, debimos detener el desarrollo para realizar pequeños ajustes sobre la misma y así continuar sobre un diseño firme.

6) Si se quiere utilizar distintos algoritmos para data science entonces se necesita tener claro cómo funcionan estos para saber qué formato esperan los datos de antemano.

Durante el proyecto nos ocurrió que avanzamos con la implementación de algunos algoritmos de clasificación de usuarios (sobre todo en las pruebas de concepto) y debimos realizar retrabajos y esfuerzo extra debido a que no teníamos del todo claro cuáles eran los formatos de entradas de datos.

7. Conclusiones

Una vez finalizado el proyecto académico y habiendo entregado el producto al cliente, podemos enumerar las siguientes conclusiones.

7.1. Conclusiones del proyecto

1) Finalizar y aprobar el proyecto de Ingeniería en setiembre de 2021 con una calificación satisfactoria para el equipo.

Para nosotros este objetivo está completado en un 90% ya que finalizamos el proyecto para setiembre de 2021 pero al momento de redactar este documento no conocemos la calificación obtenida.

2) Elaborar un producto que cumpla con los requerimientos acordados con el Cliente y entregárselo en el mes de septiembre de 2021.

En cuanto a este punto podemos decir que el objetivo se encuentra completado en un 70%, ya que, si bien cumplimos con los requerimientos funcionales y no funcionales acordados con el cliente, no llegamos a evaluar si el incremento de la métrica de frecuencia de compra fue esperado por el negocio durante el tiempo del proyecto académico. Dicha medición, como comentamos anteriormente en este documento, será a largo plazo luego de la puesta en producción del sistema.

3) Aplicar conocimientos de gestión y tecnologías aprendidos durante la carrera en beneficio de alcanzar los objetivos de producto.

Consideramos que este objetivo **ha sido cumplido en su totalidad,** ya que, además de aplicar las metodologías y conocimientos aprendidos en la carrera hemos logrado un manejo de tal nivel que nos permitió combinarlos para sacar el mayor provecho de cada uno de ellos. Además hemos investigado nuevas tecnologías y metodologías por fuera de lo aprendido en la facultad, por lo que nos encontramos ampliamente satisfechos en relación a este punto.

4) Ganar experiencia y conocimiento interactuando un año con un cliente real, pudiendo obtener aprendizajes sobre sus exigencias, modo de trabajo y conocimiento del negocio.

Consideramos que este objetivo **ha sido cumplido en su totalidad**, debido a que hemos adquirido experiencia en la interacción con un cliente real, desde una charla de presentación de una vaga idea, pasando por la investigación, el diseño e implementación de un producto en conjunto con expertos de negocio y técnicos. Además, consideramos que hemos cumplido el objetivo de desarrollar habilidades de promoción del producto desarrollado y dar visibilidad al resto de la empresa para defenderlo.

5) Dar visibilidad del producto elaborado al resto de Mercado Libre, dando a conocer las herramientas y convirtiendo a los integrantes de la misma en referentes.

Creemos que **hemos cumplido con este objetivo** en su totalidad, debido a que hemos interactuado y dado a conocer el proyecto a al menos cuatro proyectos dentro de Mercado Libre (Engagement, Acquisition, Attribution y Fraude), además de que nuestro cliente nos ha puesto en contacto con otros colegas de negocio para que le informemos acerca del proyecto, lo cual nos hace pensar que nos convirtió en referentes del mismo.

7.2. Otras conclusiones

1) Cada metodología de trabajo aplica para cada contexto y equipo. Tomemos lo mejor de ellas.

Como equipo creemos que durante el desarrollo del proyecto hemos aprendido mucho sobre metodologías de trabajo, incluso hemos llegado a descubrir nuevas metodologías más apropiadas para cada parte del proyecto. También aprendimos que es posible combinarlas (como el caso de CRISP-DM + Kanban o SCRUMBAN) y así sumar los beneficios de varias metodologías.

2) La comunicación con el equipo es fundamental para salir adelante en los problemas y dar visibilidad.

En ocasiones durante el tiempo que dedicamos al proyecto hemos tenido altibajos motivacionales y bloqueos que no nos permitían avanzar (lo cual consideramos que no es lo más saludable para un equipo de proyecto). En este equipo hemos siempre priorizado la comunicación entre integrantes, dando visibilidad al cliente y al tutor. Creemos que la comunicación y el hecho de poder abrirse a la otra persona y expresar libremente los pensamientos, críticas e ideas (siempre con el enfoque constructivo) ha ayudado a minimizar las desmotivaciones, retroalimentar al otro integrante y destrabar los bloqueos existentes.

3) La cantidad de miembros en un equipo juega un papel importante para el proyecto.

Creemos que si el equipo hubiese estado conformado por más integrantes se hubiera podido alcanzar un producto con mayor cantidad de features en producción y hubiésemos sorteado de una manera más sencilla los obstáculos que se fueron presentando. Igualmente, y como dijimos en la presentación del equipo, el hecho de existir confianza generada desde hace varios años entre nosotros como equipo de proyecto, hizo que nos potenciáramos mutuamente y realizamos un esfuerzo extra por lograr colmar las expectativas acordadas con el cliente. De todos modos, entendemos que es indudable que si el equipo hubiese estado conformado por más integrantes, el trabajo no se hubiese tornado tan pesado para cada uno de nosotros.

4) Conocer claramente la temática del proyecto que se encara es fundamental para el éxito del mismo.

Esta conclusión es muy importante para nosotros, debido a que el hecho de no disponer de un conocimiento de antemano sobre las temáticas requeridas para el desarrollo del proyecto nos sacó de nuestra zona de confort y, aunque nos permitió crecer profesionalmente, también nos llevó a momentos de frustración y a creer que sería imposible completar el proyecto exitosamente en el tiempo pautado (lo cual no ocurrió).

8. Next Steps

Una vez finalizado el proyecto, definimos los próximos pasos a seguir en una etapa post-entrega académica. Debido a que el alcance del proyecto fue reducido por disponer de un tiempo acotado, acordamos con el cliente una serie de nuevas funcionalidades e integraciones de este sistema con otros del ecosistema de Mercado Libre.

Los pasos a seguir serán los siguientes:

1) Incorporar las visitas de los usuarios en la plataforma para el algoritmo de recomendaciones: modificar el algoritmo de identificación de perfiles de usuario para que considere las visitas de los usuarios a productos de interés y no solamente utilice la información histórica de compras.

2) Priorización de usuarios según frecuencia de compra para optimización de envíos de campañas: optimizar el envío de campañas publicitarias con productos recomendados, seleccionando y apuntando a usuarios que generen un mayor retorno de inversión (a través de la frecuencia de compra).

3) Integración con más equipos para poder crear campañas más personalizadas, por ejemplo en facebook, google, etc.: construir una integración con otros sistemas para aumentar el scope del proyecto enviando publicidad a través de otros canales como facebook, google, etc.

4) Integración de las recomendaciones en el frontend de Mercado Libre: construir una integración para reutilizar el modelo de recomendaciones para mostrarlas en el sitio principal de Mercado Libre y así mejorar la experiencia de usuario, ofreciendo recomendaciones al usuario al momento de navegar el sitio.

9. Referencias bibliográficas

1- Wikipedia, “Scrum desarrollo de software”, julio 2021. [Online]. Available: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software). [Accessed: 30-Sep-2021].

2- Wikipedia, “Contributors to Wikipedia projects”, agosto 2021. [Online]. Available: https://es.wikipedia.org/wiki/Kanban. [Accessed: 30-Sep-2021].

3- Wikipedia, “Desarrollo guiado por comportamiento”, febrero 2021. [Online]. Available: https://es.wikipedia.org/wiki/Desarrollo_guiado_por_comportamiento. [Accessed: 30-Sep-2021].

4- Wikipedia, “Cross Industry Standard Process for Data Mining”, octubre 2021. [Online]. Available: https://es.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining. [Accessed: 30-Sep-2021].

5- Prem Kumar, “Netflix Movie Recommendation Using Collaborative Filtering”, marzo 2020. [Online]. Available: https://towardsdatascience.com/tensorflow-for-recommendation-model-part-1-19f6b6dc207d.[Accessed: 30-Sep-2021].

6- Wikipedia, “Modelo de vistas 4 + 1”, septiembre 2021. [Online]. Available: https://es.wikipedia.org/wiki/Modelo_de_Vistas_de_Arquitectura_4%2B1. [Accessed: 30-Sep-2021].

7- Wikipedia, “Institute of Electrical and Electronics Engineers”, junio 2021. [Online]. Available: https://es.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers. [Accessed: 30-Sep-2021].

8- Wikipedia, “Cluster analysis”, septiembre 2021. [Online]. Available: https://en.wikipedia.org/wiki/Cluster_analysis. [Accessed: 30-Sep-2021].

9- Wikipedia, “K-medias”, septiembre 2021. [Online]. Available: https://es.wikipedia.org/wiki/K-medias. [Accessed: 30-Sep-2021].

10- Wikipedia, “Collaborative filtering”, septiembre 2021. [Online]. Available: https://en.wikipedia.org/wiki/Collaborative_filtering [Accessed: 30-Sep-2021].

11- Wikipedia, “Scikit-learn”, noviembre 2021. [Online]. Available: https://es.wikipedia.org/wiki/Scikit-learn [Accessed: 30-Sep-2021].

12- Wikipedia, “Descomposición en valores singulares”, mayo 2021. [Online]. Available: https://es.wikipedia.org/wiki/Descomposici%C3%B3n_en_valores_singulares [Accessed: 30-Sep-2021].

13- Wikipedia, “Protocolo de transferencia de hipertexto”, septiembre 2021. [Online]. Available: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto [Accessed: 30-Sep-2021].

14- Wikipedia, “Kanban Board”, septiembre 2021. [Online]. Available: https://en.wikipedia.org/wiki/Kanban_board [Accessed: 30-Sep-2021].

10. Anexos

En esta sección del documento presentamos información adicional y documentos que permiten al lector ampliar detalles sobre el proyecto.

Anexo 1: Product Backlog

Requerimientos funcionales (User Stories)

US1 - Identificación de perfiles de usuario

Narrativa

Como usuario Manager de marketing de Mercado Libre
Quiero ver los perfiles de los usuarios del sistema
Para ofrecerles productos que puedan ser de su interés y los compren.

Criterios de aceptación

Escenario 1: Agrupamiento de usuarios

Dado un conjunto de usuarios de Mercado Libre que compraron artículos (1) en la plataforma
Cuando se solicita reporte de perfiles de usuarios
Entonces se muestra el último agrupamiento de usuarios, generado en base a similitud de perfiles (2).

(1) artículos comprados en una ventana de tiempo de 6 meses y que se hayan realizado en al menos 10 categorías diferentes.
(2) dos perfiles de usuario son similares si éstos han comprado en las mismas categorías, bajo las condiciones especificadas en (1).

Comentarios

Sin comentarios

US2 - Manejo de roles

Narrativa

Como sistema de control de permisos del sistema e-profiler
Quiero validar permisos a los usuarios registrados
Para que los usuarios realicen un manejo adecuado del sistema.

Criterios de aceptación

Escenario 1: Usuario manager

Dado un usuario registrado en el sistema como manager
Cuando accede al sistema e-Profiler
Entonces el sistema de permisos valida que tenga acceso a todas las funcionalidades.

Escenario 2: Usuario analista de datos

Dado un usuario registrado en el sistema como analista de datos
Cuando accede al sistema e-Profiler
Entonces el sistema de permisos valida que tenga acceso sólo a la sección de visualizar perfiles.

Escenario 3: Usuario operador de marketing

Dado un usuario registrado en el sistema como operador de marketing
Cuando se accede al sistema e-Profiler
Entonces el sistema de permisos valida que tenga acceso sólo a la sección de visualización de performance de campañas enviadas a los clientes.

Comentarios

Sin comentarios

US3 - Envío de recomendaciones automáticas a usuarios

Narrativa

Como sistema de envío de recomendaciones de e-Profiler
Quiero enviar emails con productos recomendados (a través de la integración con el sistema Hermes) a los usuarios de los perfiles identificados a través
Para ofrecer productos recomendados e intentar aumentar la frecuencia de compra de estos usuarios.

Criterios de aceptación

Escenario 1: Existen notificaciones pendientes de envío a usuarios

Dada una lista de categorías de productos a recomendar a usuarios de un perfil determinado
Cuando se ejecuta el envío automático de notificaciones
Entonces el sistema elige los 10 ítems más relevantes entre las categorías a recomendar, procesa la solicitud de envío de notificaciones al usuario (a través del sistema Hermes) y registra el envío para futuras validaciones.

Escenario 2: No hay notificaciones pendientes de envío a usuarios

Dada una lista de categorías de productos a recomendar a usuarios de un perfil determinado y existe al menos una notificación previa enviada en los últimos 7 días
Cuando se ejecuta el envío automático de notificaciones
Entonces el sistema e-Profiler no envía solicitud de generar nuevas notificaciones al sistema Hermes.

Escenario 3: Error al enviar notificaciones al usuario

Dada una lista de categorías de productos a recomendar a usuarios de un perfil determinado
Cuando se ejecuta el envío automático de notificaciones y falla la comunicación con el sistema Hermes
Entonces el sistema e-Profiler deja las notificaciones en estado ERROR para reintentar y procesarlas en el futuro.

Comentarios

Sin comentarios

US4 - Visualización de información de envíos de publicidad

Narrativa

Como usuario Manager u Operador del equipo de Marketing de Mercado Libre
Quiero visualizar información de los envíos automáticos de publicidad que se hicieron a los usuarios con productos recomendados por e-profiler
Para entender a cuántos usuarios se llegó con la publicidad y en qué fechas.

Criterios de aceptación

Escenario 1: Visualización de listado no vacío de notificaciones

Dado un usuario Manager u Operador de Marketing previamente autenticado en el sistema e-Profiler y un conjunto no vacío de notificaciones previamente enviadas
Cuando solicita ver información de notificaciones enviadas en un rango de fechas
Entonces el sistema muestra información de las notificaciones enviadas, mostrando la fecha, el estado de las mismas y la cantidad de usuarios afectados.

Escenario 2: Visualización de listado vacío de notificaciones

Dado un usuario Manager u Operador de Marketing previamente autenticado en el sistema e-Profiler y un conjunto vacío de notificaciones previamente enviadas
Cuando solicita ver información de notificaciones enviadas en un rango de fechas
Entonces el sistema muestra un listado vacío de notificaciones y un mensaje informativo.

Escenario 3: Visualización de listado no vacío de notificaciones, usuario con rol inválido

Dado un usuario que no tiene rol Manager u Operador de Marketing, previamente autenticado en el sistema e-Profiler y un conjunto no vacío de notificaciones previamente enviadas
Cuando solicita ver información de notificaciones enviadas en un rango de fechas
Entonces el sistema indica al usuario que se requiere autenticación para esta funcionalidad.

Comentarios

Sin comentarios

US5 - Generación y envío de campañas publicitarias manuales

Narrativa

Como usuario Operador de Marketing de Mercado Libre
Quiero seleccionar publicaciones recomendadas y generar campañas publicitarias en forma manual
Para enviar a usuarios de un perfil en particular para intentar aumentar su frecuencia de compra.

Comentarios

Fuera del alcance del proyecto académico.

US6 - Gestión de accesos al sistema

Narrativa

Como usuario Manager del equipo de Marketing de Mercado Libre
Quiero administrar los accesos de mis colaboradores
Para que utilicen el sistema en forma adecuada.

Comentarios

Fuera del alcance del proyecto académico.

US7 - Configuración del módulo de identificación de perfiles

Narrativa

Como usuario Manager del equipo de Marketing de Mercado Libre
Quiero poder configurar los parámetros del módulo de identificación de perfiles (1)
Para adaptar la generación de perfiles a los constantes cambios que surgen en la compañía.

(1) Los parámetros a ajustar serán los siguientes: ventana de tiempo, categorías a considerar, cantidades mínimas de compras y frecuencia de actualización de la información.

Comentarios

Fuera del alcance del proyecto académico.

US8 - Configuración del módulo de productos recomendados

Narrativa

Como usuario Manager del equipo de Marketing de Mercado Libre
Quiero poder configurar los parámetros del módulo de productos recomendados (1)
Para personalizar las recomendaciones generadas en función de las necesidades de la compañía.

(1) Los parámetros a configurar serán los siguientes: cantidades de sugerencias por perfil y ventana de tiempo a tener en cuenta.

Comentarios

Fuera del alcance del proyecto académico.

Requerimientos no funcionales

RNF1 - Interoperabilidad: todas las aplicaciones que compongan la solución del proyecto deberán interoperar dentro del ecosistema integrado por todos los sistemas de Mercado Libre.

RNF2 - Confidencialidad: se deberá respetar la privacidad de datos de los clientes de la compañía, ocultando información sensible de los mismos (nombre, documento, etc) y además no divulgar ni incluir en el código de la solución información sensible de acceso recursos privados (por ejemplo conexiones a bases de datos, servidores remotos, etc).

RNF3 - Stack tecnológico: debido a que estamos trabajando en el entorno de desarrollo de Mercado Libre, es obligatorio apegarse al stack tecnológico utilizado por la empresa (estándar definido por Fury).

RNF4 - Look and feel: se deberá utilizar el layout de la compañía para las aplicaciones de frontend que se desarrollen.

RNF5 - Acceso a la aplicación: la herramienta deberá ser accesible para operadores y supervisores de marketing de Mercado Libre. Esta herramienta estará disponible dentro de la red interna privada de la empresa.

RNF6 - Estándares de calidad: la solución deberá cumplir con los estándares de calidad estipulados por la empresa en el proceso automatizado de integración contínua (requisitos de coverage, checks de linters, estándares, etc).

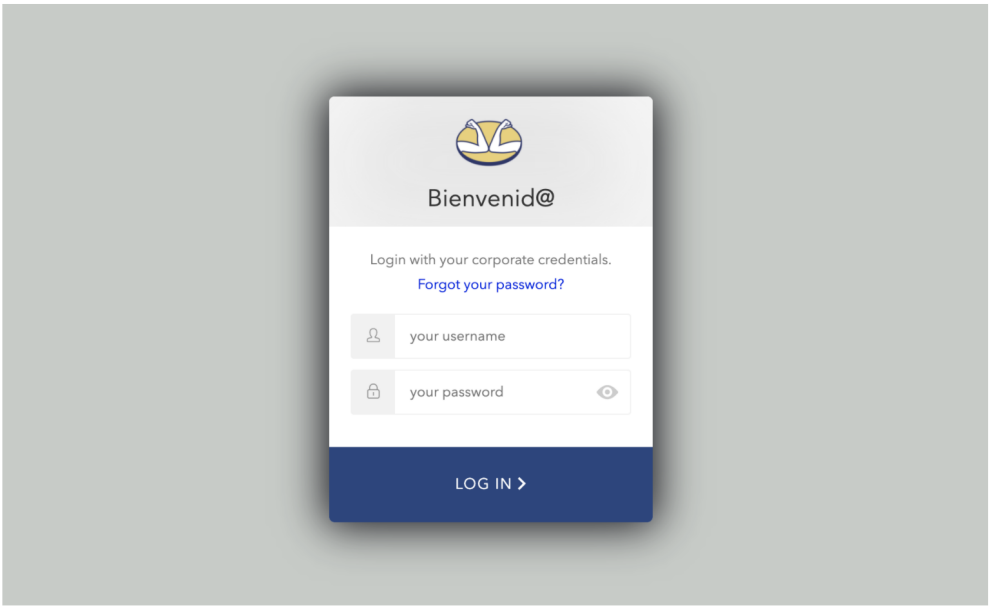
RNF7 - Utilización de los recursos de infraestructura de la compañía: el proyecto deberá funcionar sobre la infraestructura de cloud de Mercado Libre.

RNF8 - Especificaciones técnicas: El módulo de identificación de perfiles requerirá funcionar sobre una infraestructura con especificaciones como mínimo de 40 vCPU’s, 160Gb RAM y 50Gb de almacenamiento.

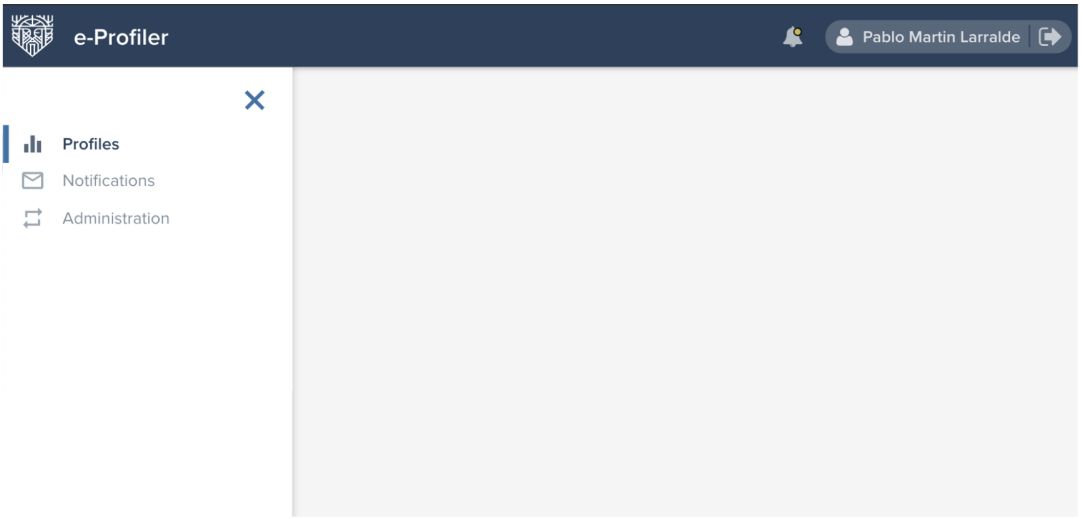
Anexo 2: Mockups de frontend

En este anexo presentamos los mockups (bosquejos livianos del software) que utilizamos esencialmente para validar requerimientos con el cliente en etapas tempranas del proyecto.

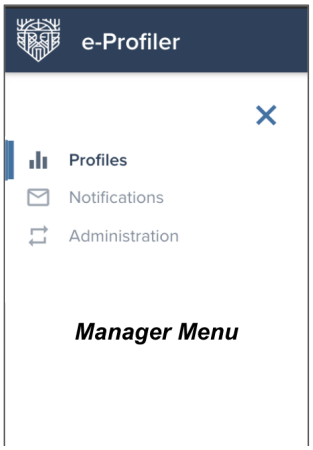
e-Profiler Login section

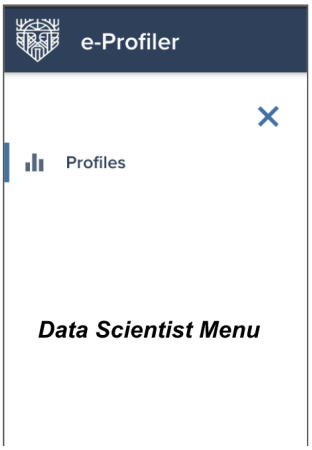


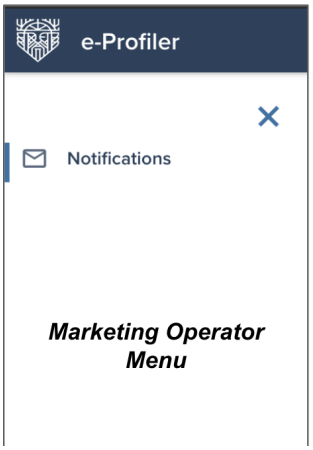
Main menu



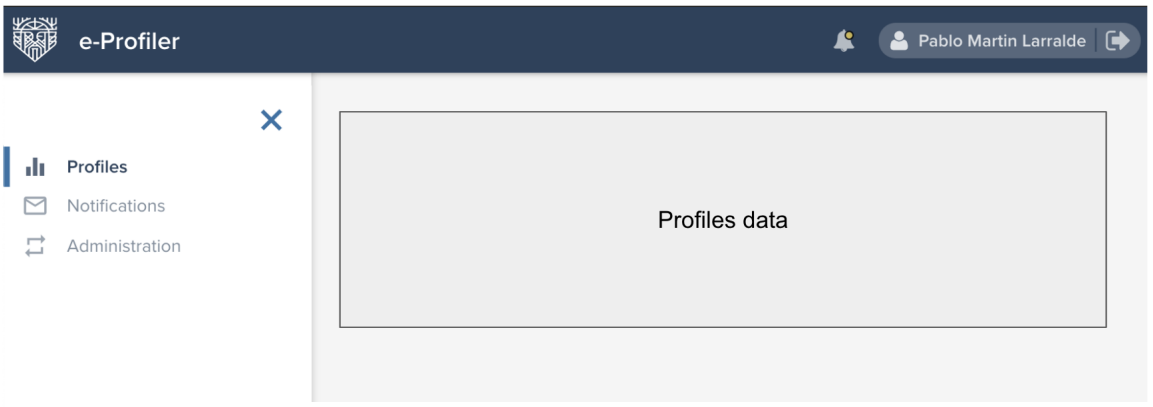
Menu options by role



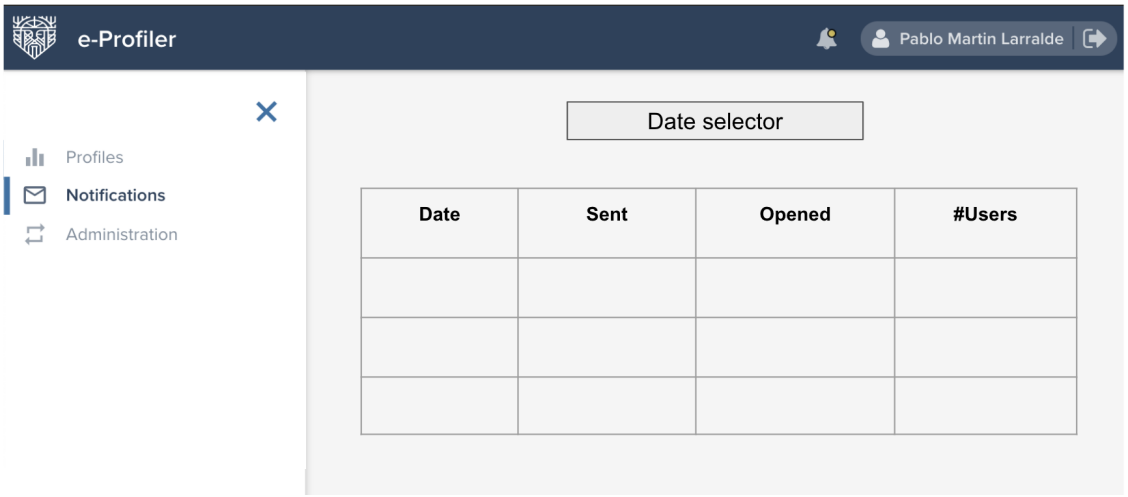




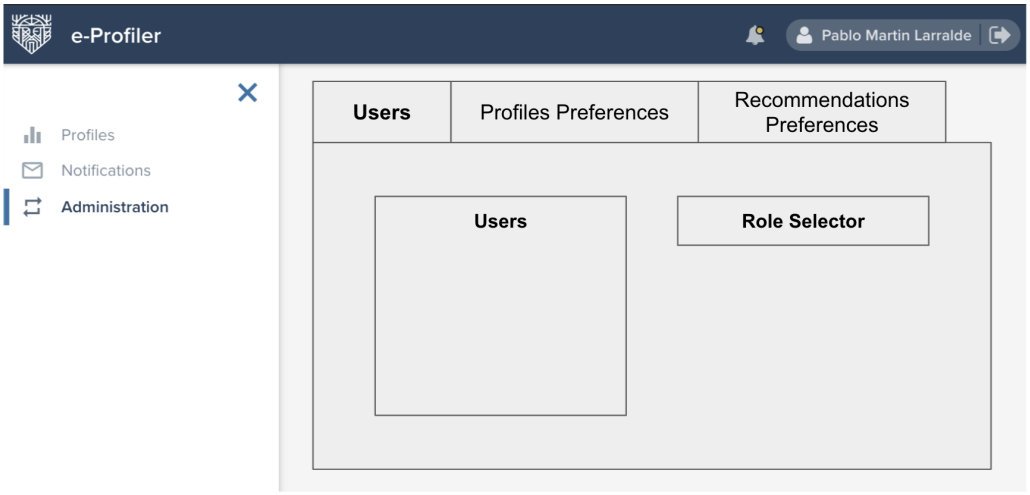
Profiles section



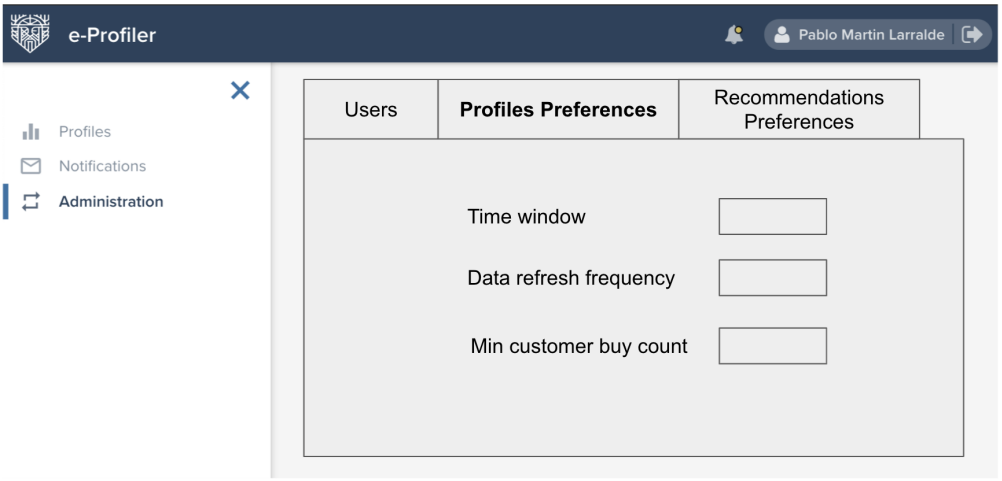
Notifications section



Users & Roles configuration



Profile discovery preferences



Recommendations preferences

Anexo 3: Planificación de riesgos

A continuación presentamos el listado de riesgos identificados en este proyecto y el plan de respuesta que determinamos para cada uno de ellos.

Riesgo 1: No manejar los conocimientos necesarios para desarrollar los módulos de data mining. Dado que el equipo de desarrollo no disponía de la suficiente experiencia ni conocimientos avanzados sobre desarrollo de proyectos de data mining. Debido a lo anterior, cuando tuvimos que implementar los módulos de data mining, entendimos que podría tener un gran impacto en el alcance, tiempo y costo del proyecto, ya que consideramos que los módulos de data mining son de especial interés para el cliente del proyecto.

Plan de respuesta: Capacitar al equipo en las tecnologías necesarias para desarrollar el proyecto, principalmente Machine Learning. Disponer de un referente (líder técnico) a quien consultar periódicamente y evitar que este riesgo tenga alto impacto.

Riesgo 2: Falta de experiencia del equipo para gestionar un proyecto mayoritariamente de data mining. Dado que como equipo no disponía de experiencia para gestionar un proyecto que en su mayoría trata de módulos de data mining, además tampoco se contó con un amplio manejo de las metodologías más apropiadas para este tipo de proyecto y cuando tuviésemos que gestionar este tipo de proyecto, nos podría impactar en el alcance, tiempo y costo del proyecto, ya que consideramos que los módulos de data mining son de especial interés para el cliente del proyecto.

Plan de respuesta: Basarnos en ejemplos de proyectos similares que hayan sido exitosos, consultar a expertos en este tipo de proyectos y combinar metodologías para lograr cumplir con los objetivos del proyecto y las expectativas del cliente.

Riesgo 3: El alcance del proyecto es tan amplio que no puede ser completado por un equipo de dos integrantes en el tiempo estipulado por la Universidad ORT Uruguay. Dado que como nuestro equipo de proyecto estuvo conformado únicamente por dos integrantes, y cuando evaluamos el alcance planteado por el cliente, existía la posibilidad de que no pudiésemos completar las tareas requeridas.

Entonces entendimos que nos podría impactar en el alcance, tiempo y costo del proyecto, ya que deberíamos acordar un ajuste del alcance del proyecto con el cliente para cumplir con el requisito académico de entregar el proyecto el 30 de setiembre de 2021.

Plan de respuesta: Estimar las tareas necesarias para cumplir con el proyecto e ir comparando los totales con valores alcanzables de esfuerzo, considerando las tareas fuera del proyecto que realizan los integrantes. Evaluar periódicamente el impacto de este riesgo y de ser necesario, re-negociar con el cliente y priorizar las funcionalidades que le agregan más valor, difiriendo las menos prioritarias para etapas posteriores a la entrega del proyecto académico.

Riesgo 4: El proceso de ETL insume demasiado tiempo y retrasa todo el flujo de identificación de perfiles y recomendaciones. Dado que nuestro proyecto se trató fundamentalmente de operaciones de data mining, el proceso de ETL cumplió un papel fundamental dentro del mismo, entonces entendimos que si la ejecución del proceso de ETL insumió más tiempo del requerido para obtener resultados en el tiempo esperado por el cliente, entonces se retrasaría todo el flujo de identificación de perfiles y recomendaciones.

Plan de respuesta: Hacer mediciones tempranas de la cota superior del tiempo que insumiría el proceso de ETL y determinar si es necesario segmentarlo.

Riesgo 5: El resultado del identificación de perfiles y recomendaciones no es de utilidad para el negocio. Dada la incertidumbre que se manejaba al momento del planteamiento de la necesidad del cliente y el hecho de trabajar con tecnologías y metodologías poco conocidas por nosotros como equipo de desarrollo, una vez que entreguemos los resultados al cliente, existía la posibilidad de que el resultado de la identificación de perfiles y las recomendaciones no fueran de utilidad para el negocio, o lo fueran pero en un porcentaje que no marque una diferencia con el modelo actual de recomendaciones. Entendiendo que este es un módulo central del proyecto, si este riesgo se materializa, el impacto se daría en los parámetros de tiempo y costo del proyecto, ya que deberíamos replantearnos la forma de implementarlo y luego volver a iterar con el cliente.

Plan de respuesta: Utilizar algoritmos conocidos de análisis de datos (no reinventar la rueda), reunirnos frecuentemente con expertos sobre el tema, además de presentar avances al cliente en cada release, de modo de evaluar la desviación.

Riesgo 6: No se cumple el principio de integridad de los datos a utilizar. Dado que estamos extrayendo y utilizando información de diferentes fuentes de datos, cuando realizamos el procesamiento de datos para la identificación de perfiles y recomendaciones, entonces es posible que se produzcan inconsistencias eventuales en los mismos que afecten la calidad del producto final y consecuentemente la satisfacción del cliente. Esto también podría afectar el alcance y duración del proyecto, debido a que sería necesario adoptar medidas adicionales para asegurar la integridad de los datos.

Plan de respuesta: Reducir al mínimo la utilización de orígenes de datos y basarnos solamente en las tablas centrales de Mercado Libre (las cuales contienen datos que tienen integridad referencial por la propia supervivencia de la empresa).

Riesgo 7: El equipo no dispone de los conocimientos requeridos para la integración con el sistema de envíos de notificaciones. Dado que nuestra solución debe integrarse con un sistema externo para el envío de notificaciones, cuando realicemos dicha integración entonces es posible que no contemos con los conocimientos necesarios para implementarla. Esto podría afectar el alcance y duración del proyecto, debido a que requerimos soporte técnico y de negocio para dejar la funcionalidad operativa.

Plan de respuesta: Recurrir a expertos referentes del equipo de Hermes (sistema de notificaciones), capacitarnos como equipo con documentación escrita y audiovisual disponible. Solicitar apoyo de un referente del equipo de marketing y notifications para que nos brinde soporte en cuanto a las tecnologías y a los asuntos de negocio.

Riesgo 8: La gestión de accesos a los recursos restringidos de la compañía insumen demasiado tiempo. Dado que estamos inmersos en el ecosistema de aplicaciones y recursos de la empresa Mercado Libre, parte de los procesos de autorización para acceder a dichas aplicaciones y recursos requieren el cumplimiento de pasos estrictos que demandan interacción de varias personas.

Cuando sea necesario acceder a algún recurso o aplicación que interactúe con nuestro proyecto, entonces es posible que dichos procesos demanden demasiado tiempo y ésto haga que estos tiempos impacten negativamente en el tiempo y alcance del proyecto.

Plan de respuesta: para agilizar estos tiempos y evitar que se alarguen mucho los tiempos de ejecución del proyecto debido a solicitudes de permisos, involucramos a líderes y supervisores de equipos que puedan ayudarnos a destrabar estos temas.

Riesgo 9: Indisponibilidad del cliente para reunirse con el equipo y aportar valor al producto. Dado que nuestro cliente es una persona con muchas responsabilidades y está constantemente ocupado con cuestiones de negocio de alta gerencia, cuando necesitemos su atención para cuestiones relacionadas con el proyecto (consultas, validación de requerimientos, feedback sobre el producto, etc) entonces es posible que no disponga de tiempo suficiente para dedicarnos. Ésto tendrá un impacto negativo en el tiempo del proyecto, debido a que retrasaría toda la ejecución.

Plan de respuesta: Utilizar mecanismos alternativos de comunicación con el cliente, de modo de no perder contacto y evitar el alejamiento del proyecto, enviar comunicaciones (chats) cada poco tiempo para ir validando sus necesidades y recogiendo feedback que pueda mejorar el producto final.

Anexo 4: User stories por release

Una vez definido el Product Backlog especificado en el Anexo 1: Product Backlog y, en base a las prioridades definidas por el cliente, realizamos la siguiente distribución de user stories por cada release.

4.1. Release 1

- US1 - Identificación de perfiles de usuario
- US2 - Manejo de roles

4.2. Release 2

- US3 - Envío de recomendaciones automáticas a usuarios
- US4 - Visualización de información de envíos de publicidad

4.3. Etapa post-proyecto académico

- US5 - Generación y envío de campañas publicitarias manuales
- US6 - Gestión de accesos al sistema
- US7 - Configuración del módulo de identificación de perfiles
- US8 - Configuración del módulo de productos recomendados

Anexo 5: Seguimiento de los releases del proyecto

En este anexo presentamos el detalle de los releases planificados para el proyecto, desde su definición hasta el seguimiento y los resultados obtenidos.

5.1. Release 1

Objetivos del release

Como acordamos con el cliente, el foco principal estuvo puesto en la identificación de perfiles de usuario, ya que es la funcionalidad de mayor prioridad para el cliente, por lo que los objetivos para el release fueron los mencionados a continuación.

Construir las siguientes funcionalidades:

- Identificación de perfiles de usuario
- Primera versión del frontend de la aplicación e-Profiler: scaffolding inicial del proyecto, menú principal y validación de permisos según roles del usuario logueado.

Planificación

- Fecha de Comienzo: 01/04/2021
- Fecha de Fin: 31/07/2021
- Capacidad planificada del equipo: 160 hs (20 horas por semana, 4 semanas y con un equipo de 2 personas)

Definición de Hecho

- Cada historia de usuario/funcionalidad está codificada, compilada y desplegada en scopes productivos de Fury.

- Las historias de usuario/funcionalidades cumplen todos los criterios de aceptación definidos y solicitados por el cliente, funcionan correctamente en el ambiente productivo y cuentan con pruebas funcionales automatizadas, las cuales luego de ejecutarlas su resultado es verde.

- No existen advertencias en rojo en la verificación de código del servidor de integración.

- Las clases de negocio cuentan con pruebas unitarias y de integración que verifican su correcto funcionamiento.
- El código desarrollado está comentado y estandarizado, además de está almacenado en el repositorio de control de versiones.
- El porcentaje de cobertura de pruebas unitarias al código es mayor o igual al estándar definido por fury (el porcentaje es superior a un 80%)
- El proceso de ETL y training para identificación de perfiles se encuentra definido, configurado y funcionando sobre el ambiente de fury.

Planificación y seguimiento de riesgos

Como comentamos anteriormente en el documento, en la release planning meeting identificamos los riesgos del release backlog que aplicaban para cada release y establecimos planes de respuesta para cada uno de ellos.

Otra de las actividades realizadas al finalizar el release fue la revisión de riesgos, cuyo resultado presentamos a continuación.

Revisión de riesgos (Fecha: 15-08-2021)

1) No manejar los conocimientos necesarios para desarrollar los módulos de data mining

El riesgo desapareció, dado que adoptamos la estrategia de reunirnos frecuentemente con un experto en data mining de Mercado Libre, quien nos orientó y nos sugirió las mejores formas de resolver el problema planteado. (Fecha de resolución aproximada: 13 de junio de 2021).

2) Falta de experiencia del equipo para gestionar un proyecto mayoritariamente de data mining

El riesgo desapareció a partir de las sugerencias que recibimos en relación a la metodología más apropiada para gestionar proyectos de este tipo. Luego a partir de las sugerencias y la investigación realizada, adoptamos la metodología CrispDM y la adaptamos a nuestro proyecto, generando una metodología que nos permitió continuar con el desarrollo del mismo. (Fecha de resolución aproximada: 15 de mayo de 2021)

3) El alcance del proyecto es tan amplio que no puede ser completado por un equipo de dos integrantes en el tiempo estipulado por la Universidad ORT Uruguay.

Este riesgo se convirtió en problema, debido al desconocimiento e inexperiencia que teníamos como equipo en relación al desarrollo y gestión de un proyecto de data mining. Por esta razón debimos aplicar el plan de contingencia, el cual consistía en re-negociar con el cliente y priorizar las funcionalidades que le agregan más valor, difiriendo las menos prioritarias para etapas posteriores a la entrega del proyecto académico. Por esta razón decidimos incluir dos releases para este proyecto académico y diferir las funcionalidades que correspondían al tercero (aquellas que no son tan prioritarias para el cliente) para futuras versiones del sistema, luego de la entrega del proyecto académico.

4) El proceso de ETL insume demasiado tiempo y retrasa todo el flujo de identificación de perfiles y recomendaciones.

Este riesgo desapareció en el momento que logramos (con la infraestructura de Mercado Libre - FDA) que el proceso de ETL se completara en un tiempo menor al umbral mínimo establecido, aún con una ventana de tiempo mayor a la estimada inicialmente. Para ver los valores de tiempos, referirse a la sección de métricas. (Fecha de resolución aproximada: 15/07/2021)

5) El resultado del identificación de perfiles y recomendaciones no es de utilidad para el negocio

Este riesgo desapareció una vez que logramos definir una métrica a largo plazo con el cliente para poder medir la performance del proyecto. Esto nos permitirá medir en qué medida el análisis realizado es útil para el cliente y el negocio, además de darnos la pauta de que vamos en la dirección correcta. (Fecha de resolución aproximada: 26/07/2021).

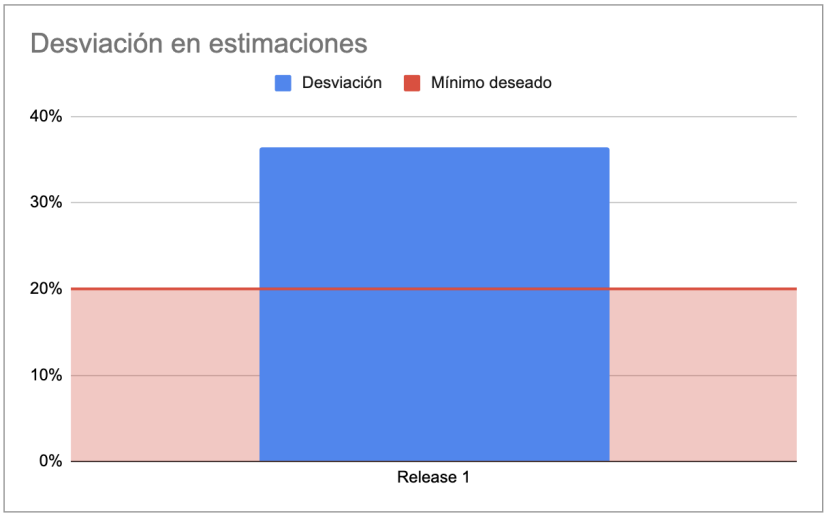
6) No se cumple el principio de integridad de los datos a utilizar

El riesgo desapareció en la medida que los orígenes de datos que elegimos para los análisis son los mínimos necesarios y son los utilizados por Mercado Libre para sus operaciones diarias, por lo que se presumen consistentes (Fecha de resolución aproximada: 01/06/2021).

Seguimiento de métrica de desviación en estimaciones y distribución del esfuerzo

Una vez finalizado el release analizamos la métrica de desviación en las estimaciones respecto del plan, para este caso el valor de la desviación promedio fue de 36%, un 16% por encima del mínimo deseado por el equipo de 20%.

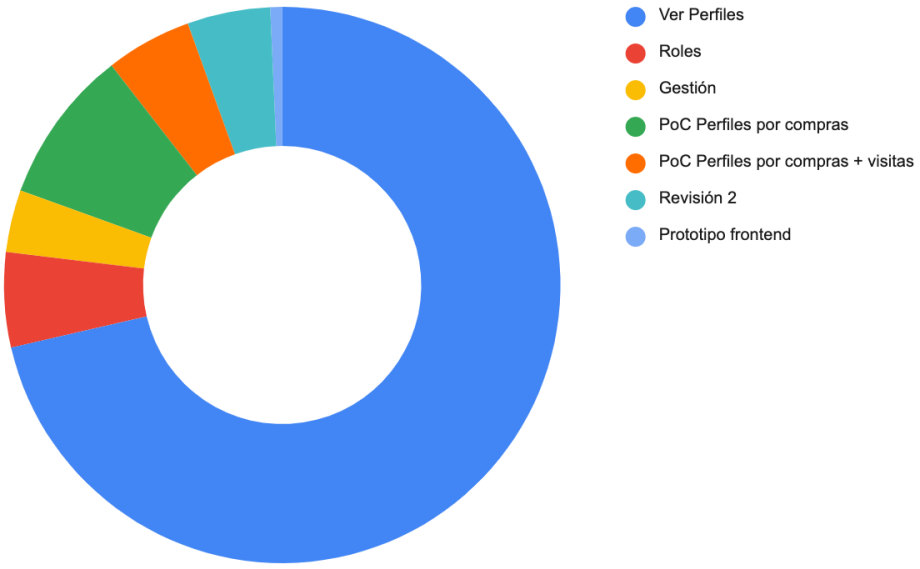
	Desviación	Mínimo deseado
Release 1	36%	20%



A continuación presentamos las estadísticas de estimaciones, junto con el % de desviación de cada una de las tareas.

US	SP	Piece	Estimado en HS	Real en HS	% desviación
	8	Frontend	38	41	8%
Ver Perfiles	13	Backend	50	52	4%
	34	AI	164	321	96%
	3	Frontend	16	20	25%
Roles	2	Backend	10	12	20%
Gestión	3	-	20	21	5%
PoC Perfiles por compras		AI	52	52	0%
PoC Perfiles por compras + visitas		AI	29	29	0%
Revisión 2			40	28	-30%
Prototipo frontend		Frontend	6	4	-33%
TOTAL	**63**	**-**	**425**	**580**	**36,47%**

En el siguiente gráfico se muestra la distribución de esfuerzo del equipo a lo largo del release.



Si analizamos el cuadro y el gráfico anterior, vemos que el mayor porcentaje de desviación se dio en las tareas del módulo de identificación de perfiles (Artificial Intelligence). Aquí vemos materializado el riesgo relacionado con la falta de conocimiento del equipo en relación a las tecnologías de proyectos de data mining.

En este caso, además del plan de respuesta, pusimos en práctica el plan de contingencia al darnos cuenta que el riesgo se convirtió en problema. Las acciones tomadas resultaron en que desarrollamos un spike que incluyó dos pruebas de concepto (PoC de perfiles por compras y PoC de perfiles por compras + visitas), las cuales nos permitieron destrabar el proyecto y, en conjunto con el cliente, decidimos avanzar con ese rumbo. Como estas pruebas de concepto no estaban planificadas dentro del scope del release, la estimación fue realizada post-mortem (por esta razón la desviación es de 0%).

Las pruebas de concepto mencionadas las realizamos con la colaboración de expertos en data science de la empresa.

Debido a la situación que se dio, los spikes realizados y las acciones que tuvimos que tomar para poder continuar con el proyecto, tuvimos un retraso en la planificación que nos modificó el cronograma del proyecto.

5.2. Release 2

Una vez que completamos el release 1, comenzamos con la ejecución del segundo release de la etapa de delivery del proyecto. A continuación presentamos detalles de dicha ejecución.

Objetivos del release

Continuando en la línea de tomar las tareas en el orden de prioridad establecido en conjunto con el cliente del proyecto, además de los requisitos académicos, los objetivos del release fueron los siguientes:

1. Construir las siguientes funcionalidades:

- Envío de publicidad a los usuarios con productos recomendados según su perfil.
- Visualización de información de campañas enviadas a los usuarios.

2. Preparar la tercera revisión del proyecto

Planificación

- Fecha de Comienzo: 16/08/2021
- Fecha de Fin: 15/09/2021
- Capacidad planificada del equipo: 160 hs (20 horas por semana, 4 semanas y con un equipo de 2 personas)

Definición de Hecho

- Cada historia de usuario/funcionalidad está codificada, compilada y desplegada en Fury.
- Las historias de usuario/funcionalidades cumplen todos los criterios de aceptación definidos y solicitados por el cliente, funcionan correctamente en el ambiente productivo y cuentan con pruebas funcionales automatizadas, las cuales luego de ejecutarlas su resultado es verde.
- No existen advertencias en rojo en la verificación de código del servidor de integración.
- Las clases de negocio cuentan con pruebas unitarias y de integración que verifican su correcto funcionamiento.
- El código desarrollado está comentado y estandarizado, además de está almacenado en el repositorio de control de versiones.
- El porcentaje de cobertura de pruebas unitarias al código es mayor o igual al estándar definido por fury (el porcentaje es superior a un 80%)
- El proceso de ETL y training para identificación de perfiles y recomendaciones se encuentra definido, configurado y funcionando sobre el ambiente de Fury.

Planificación y seguimiento de riesgos

Al igual que en el Release 1, realizamos la planificación de los riesgos en la release planning meeting y la revisión de los mismos en la release retrospective meeting.

A continuación presentamos un documento de revisión de riesgos realizado al finalizar el release.

Revisión de riesgos (Fecha: 20-09-2021)

7) El equipo no dispone de los conocimientos requeridos para la integración con el sistema de envíos de notificaciones.

El riesgo desapareció debido a que logramos reunirnos frecuentemente y obtener soporte sobre el funcionamiento del sistema. (Fecha de resolución aproximada: 13 de setiembre de 2021).

8) La gestión de accesos a los recursos restringidos de la compañía insumen demasiado tiempo.

El riesgo se materializó, pero no nos significó un gran impacto, ya que logramos aplicar el plan de respuesta y controlarlo. El hecho de disponer de contactos dentro de la compañía para agilizar estos procesos nos ayudó mucho para conseguir los accesos requeridos (Fecha de resolución aproximada: 12 de setiembre de 2021).

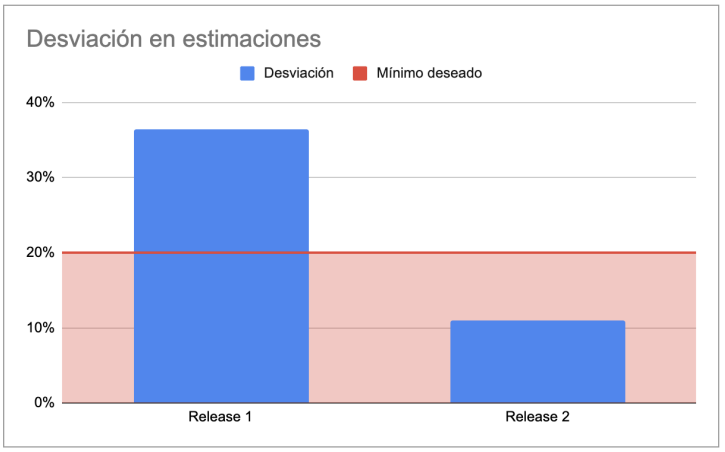
9) Indisponibilidad del cliente para reunirse con el equipo y aportar valor al producto

El riesgo se materializó y se convirtió en problema, dado que el cliente estuvo ausente por diversos motivos en varias ocasiones. Por esa razón nos vimos obligados a aplicar el plan de respuesta, el cual consistió en utilizar otros medios de comunicación alternativos a las reuniones presenciales o virtuales. La opción que elegimos fue el envío de email con el estado del proyecto y los avances del mismo. Esta opción nos fue de utilidad ya que el cliente tenía más contexto del proyecto al momento de reunirnos.

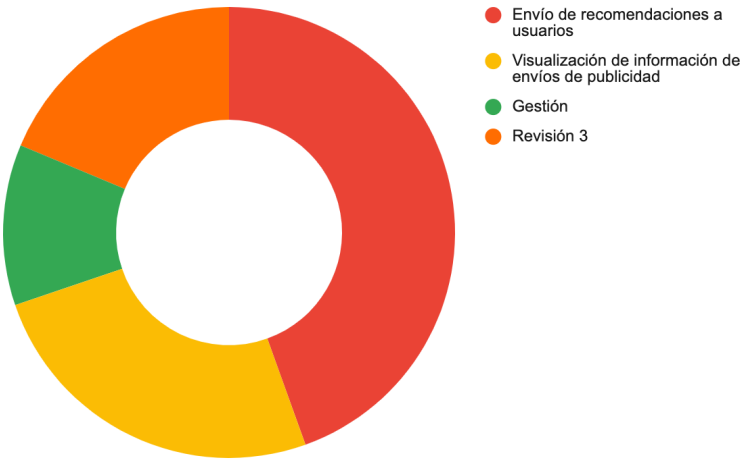
Seguimiento de la métrica de desviación en estimaciones y distribución del esfuerzo

Para continuar, analizamos la métrica de desviación en las estimaciones respecto del plan, para este release el valor de la desviación promedio fue de 11%, un 9% por debajo del mínimo deseado por el equipo de 20%.

	Desviación	Mínimo deseado
Release 1	36%	20%
Release 2	11%	20%



US	SP	Piece	Estimado en HS	Real en HS	% desviación
Envío de recomendaciones a usuarios	21	Backend	41	51	24%
	8	AI	25	30	20%
Visualización de información de envíos de publicidad	5	Frontend	18	13	-28%
	13	Backend	36	33	-8%
Gestión	3	-	14	21	50%
Revisión 3			30	34	13%
TOTAL	**50**	**-**	**164**	**182**	**10,98%**



Interpretando los resultados podemos concluir que el hecho de tomar en cuenta la desviación en las estimaciones del release 1 para las estimaciones del release 2, nos permitió bajar la desviación de este último release y mejorar la calidad de las estimaciones.

Anexo 6: Definición de métricas

En este anexo describimos las métricas consideradas en este proyecto con mayor nivel de detalle que en el documento principal. Las mismas las clasificamos como métricas de negocio, de gestión y de calidad de la solución.

De cada métrica que tuvimos en consideración detallaremos su descripción, la forma de cálculo o relevamiento, el valor real y objetivo.

6.1. Métricas de negocio

Métrica 1: Frecuencia de compra de usuarios

Refiere a la frecuencia con la que los usuarios concretan una compra en Mercado Libre en determinada ventana de tiempo. Más concretamente representa la cantidad promedio de días transcurridos entre una compra y otra.

Esta métrica fue seleccionada por nuestro cliente y será utilizada para determinar el éxito del proyecto.

Relevamiento del valor: obtenido en base a información histórica de compras realizadas en Mercado Libre y realizando los cálculos según la siguiente fórmula.

$$\text{Frecuencia de compra (FP)} = \frac{\text{Número de pedidos en (x tiempo)}}{\text{Total de número de clientes únicos en (x tiempo)}}$$

Valor objetivo: esta métrica se irá evaluando en la medida que avance el funcionamiento en producción del sistema y se produzcan variaciones en la misma (creemos que serán necesarios al menos 6 meses de funcionamiento del sistema en producción para ver variaciones). Al finalizar el proyecto, acordamos con el cliente un valor objetivo estimado de 20 días entre compras.

Métrica 2: Duración del proceso de ETL

Refiere al tiempo que insume la ejecución de las operaciones de ETL requeridas para el módulo de identificación de perfiles y recomendaciones.

Esta métrica es considerada de gran utilidad dentro del negocio, ya que de ella depende el tiempo de disponibilización de los resultados del proceso de identificación de perfiles y recomendaciones para el cliente. Por lo anterior, su valor debe ser seguido de cerca para construir una solución que esté alineada a las expectativas de negocio.

Relevamiento del valor: reportes de ejecuciones automáticas de fury.

Valor objetivo: tiempo de ejecución menor a 4 hs.

Métrica 3: Grado de satisfacción del cliente con el producto

Apreciación del cliente sobre el producto entregado al finalizar el proyecto, en términos personales y de negocio.

Esta métrica tuvo la motivación de entender en qué medida fue aceptado el producto elaborado por el equipo por parte del cliente y de esta forma poder cuantificar la calidad del trabajo realizado por nosotros.

Relevamiento del valor: feedback recibido en reuniones y resultado de encuesta de satisfacción del cliente.

Valor objetivo: calificación mayor o igual a 8 (en una escala de 0 a 10).

6.2. Métricas de gestión

Métrica 4: Desviación en estimaciones de esfuerzo por release

Refiere a la diferencia entre las estimaciones de esfuerzo en horas y las que realmente insumió una tarea.

Esta métrica fue un indicador clave para nosotros como equipo, porque debido a la gran incertidumbre manejada, necesitábamos cuantificar en qué medida lo fue. Esto con el objetivo de utilizar esta experiencia para los próximos pasos de este proyecto u otros nuevos a desarrollar.

Relevamiento del valor: registros de horas realizados por el equipo.

Valor objetivo: inferior a un 20%.

6.3. Métricas de calidad de la solución

Métrica 5: Usabilidad del frontend

Refiere a la medida de usabilidad, o facilidad de uso del frontend desarrollado para el proyecto. Para nosotros es una métrica muy importante porque nos permitió entender cuán agradable y fácil de usar es el frontend para los usuarios finales de la empresa.

Relevamiento del valor: basado en heurísticas de Nielsen, ranking de reglas.

Valor objetivo: promedio de rankings mayor a 3,5 (en una escala de 0 a 5).

Métrica 6: Porcentaje general de cobertura de código

Esta métrica de calidad se refiere al porcentaje de código que está cubierto por testing automatizado. Además de tratarse de una restricción, creemos que esta métrica fue muy importante para saber en qué medida se cumplen con las especificaciones definidas en los tests en forma automatizada.

Relevamiento del valor: reporte de cobertura de Fury.

Valor objetivo: porcentaje de cobertura mayor o igual a 80%.

Anexo 7: Evaluación de metodologías

En este anexo ampliamos información sobre la evaluación de metodologías consideradas para cada sub-proyecto dentro del proyecto principal, detallando alternativas evaluadas y sus características.

Evaluación de metodologías para el sub-proyecto de backend/frontend

Dentro de las metodologías ágiles, manejamos las siguientes alternativas:

1) Kanban

- Fácil de implementar.
- Se trata de tener un conjunto de tareas e irlas transicionando dentro de un tablero, cuyos estados pueden ser TODO, DOING y DONE.
- La idea es crear un flujo continuo e ininterrumpido, siendo lo que interesa la velocidad y la continuidad del movimiento en las tareas.

2) Scrum

- Marco de trabajo para colaboración orientado a obtener el mejor resultado posible, debido a que se va iterando el producto a medida que se van presentando prototipos al cliente.
- Permite ir teniendo entregables intermedios para ir dando visibilidad al cliente y recibir su feedback.
- Su implementación se ejecuta en ciclos temporales cortos y de duración fija (lo más común es que duren 2 semanas).
- Propone una serie de ceremonias que tienen como objetivo ir dando seguimiento al proceso y a su vez, orientar el desarrollo hacia lo que brinde mayor valor al cliente.

3) Tailored Scrum

La idea es adaptar Scrum a las necesidades del equipo, no utilizando una versión 100% purista en relación a lo que se propone en la teoría, sino que adaptarlo a las necesidades del equipo de proyecto, a la realidad del mismo y a sus stakeholders.

4) Dual track Agile

- Propone dos ciclos, uno de descubrimiento de producto y otro de entrega.
- Una parte del equipo se dedica al descubrimiento del producto, relacionándose con el cliente, de modo de ir confeccionando el backlog a medida que se va relevando.
- Otra parte del equipo se encarga de desarrollar la solución en base a lo relevado por el otro team.

Evaluación de metodologías para el proyecto de data mining

1) Cross Industry Standard Process for Data Mining (CRISP-DM): esta metodología define claramente 6 fases que deben estar presentes en un proyecto de este tipo. Estas fases son:

- Comprensión del negocio (Business Understanding)
- Comprensión de los datos (Data Understanding)
- Preparación de los datos (Data Preparation)
- Fase de modelado (Modelling)
- Evaluación (Evaluation)
- Implantación (Deployment)

2) Knowledge Discovery in Databases (KDD): es básicamente un proceso automático en el que se combinan descubrimiento y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos (data mining) y presentar resultados.

3) Sample, Explore, Modify, Model, Assess (SEMMA): Esta metodología, se centra más en el desarrollo del proceso de minería de datos y no se orienta a objetivos empresariales. Cada fase representa la inicial de las siglas que la identifican.

Anexo 8: Alternativas de collaborative filtering

En esta sección describimos con mayor nivel de detalle los tipos de collaborative filtering que investigamos para este proyecto.

1) Memory-based collaborative filtering

El enfoque memory-based o basado en memoria de su traducción al español utiliza la información de rating (valoración o puntaje) para calcular la similaridad entre los usuarios o ítems a trabajar.

En el filtrado llamado neighbor-based, donde la palabra “neighbor” proviene de su traducción del inglés en “vecino” y se basa en que los usuarios son seleccionados en base a su similitud. Esta es determinada en base al match entre usuarios con categorías de las compras que realizaron en la plataforma, de ahí la relación con el concepto de “vecino”.

Ejemplos típicos son el neighborhood-based collaborative filtering e item/user-based top N recomendaciones donde N representa un número natural positivo.

El algoritmo neighborhood-based calcula la similaridad entre más de 2 usuarios para producir como salida una predicción para el usuario tomando el peso del promedio de los ratings.

2) Model-based collaborative filtering

Los tipos basados en modelos son desarrollados usando una combinación de algoritmos de machine learning y data mining para predecir los ratings de los usuarios para ítems que no disponen ratings o puntos de valoración por parte del usuario. Existen varios algoritmos de model-based collaborative filtering como por ejemplo redes bayesianas, modelos de cluster entre otros.

Con este enfoque son comunes las técnicas de reducción de la dimensionalidad como forma complementaria para mejorar la robustez y precisión.

Recordemos que estos modelos operan con una gran cantidad de datos y una de las desventajas por ende es la gran dimensionalidad de las matrices que contienen muchos valores nulos en ella, debido a que no existen ratings entre los usuarios y todos los ítems de interés.

Una representación más reducida podría ser utilizada para operar tanto con una estrategia basada en user-based ó item-based neighborhood. Son varias las ventajas con este paradigma debido a que permite manejar mejor la dispersión de la matriz original que las memory-based. El fin de esto radica en poder calcular la similaridad de forma más performante teniendo en cuenta que la matriz es pequeña, recordando que la gran cantidad de datos involucrados en estas matrices producen matrices muy dispersas.

3) Hybrid collaborative filtering

Son varias las soluciones que combinan la estrategia memory-based y model-based.

Gracias al trabajo de ambos es posible superar las limitaciones nativas que acarrea collaborative filtering como también mejorar la performance en la predicción cuando se los necesita.

Uno de los principales problemas que se mitiga son la dispersión en las matrices de datos como también el concepto de “data loss” que refiere a una condiciones de error provocado por la pérdida de información que hace que el conjunto de datos pierdan sentido e introduzcan ruido.

Conclusiones

Entendiendo cada uno de los métodos descritos anteriormente, tratamos de obtener lo mejor de cada uno sabiendo de qué forma se podrían integrar de la mejor manera para arribar a los resultados deseados.

Como se detalló en la forma de construcción de la Matriz colaborativa, esta se componía de valores para cada una de las categorías presentes. Si bien no se hacía uso de un valor porcentual o cálculo sobre el rating de forma directa, bastó la presencia de una compra para cualquiera de las categorías presentes. Esto ya era suficiente para nosotros como para afirmar que un usuario había comprado en una

categoría. Recordemos al lector qué parte de las pre condiciones iniciales es qué el usuario hubiera realizado más de 10 compras en el site de Brasil, pero esta etapa de filtrado se realizó en el componente de ETL, pudiendo así desacoplar dichas responsabilidades de la lógica del algoritmo.

De la técnica basada en Model-Based obtuvimos la idea de poder reducir las dimensiones de las matrices, pudiendo así mejorar la performance y tiempos de respuesta cuando se ejecutaba el componente Model, donde se alojaban los algoritmos.

En el proyecto utilizamos la categoría Híbrida debido a que, si bien no tiene una descripción particular, es la que corresponde a la utilización de más de una técnica en un proyecto de Data Mining.

Una de nuestras principales estrategias es poder combinar distintas técnicas para obtener mejores beneficios.

Anexo 9: Fury Workflow (Gitflow)

Gitflow es un workflow publicado y popularizado que define un branching model estricto, diseñado con foco en lograr releases de calidad y proveé un framework orientado a la gestión de proyectos medianos y/o grandes.

Es un workflow ideal para proyectos con despliegues programados. Pero este no es un requisito excluyente para usarlo, ya que es muy útil para aplicar en equipos que tienen una periodicidad no definida de despliegues, pero que quieren garantizar un alto estándar de calidad en sus cambios.

Funcionamiento

Gitflow define una nomenclatura y roles muy específicos para cada uno de los diferentes branches y también cómo y cuándo tienen que interactuar entre sí. Por otro lado, usa branches individuales para preparar, probar y arreglar una versión determinada y así garantizar que no sea contaminada con cambios no deseados.

Branches Destacados

Los branches que forman parte de Gitflow son:

- master
- hotfix/.*
- release/.*
- develop
- feature/.*
- revert-.*
- enhancement/.*
- fix/.*
- bugfix/.*
- migration/.*

Branches Estables

Los branches estables de Git Flow son:

- master
- develop

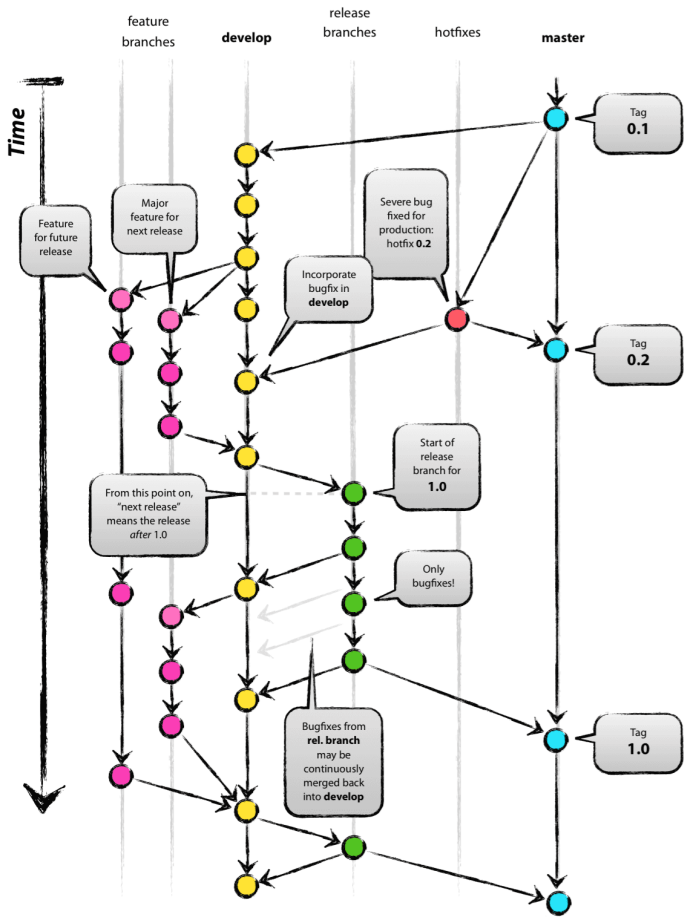
Que un branch sea estable significa que está protegido en Github y por ende no podemos hacer git push en ese branch. Las únicas interacciones posibles son a través de pull requests y para realizar el merge van a tener que cumplirse todos los controles de Release Process, que serán visibles en dicho pull request.

Los branches releseables de Gitflow son:

- master
- hotfix/.*

Que un branch sea releaseable significa que solamente desde ese branch podemos generar versiones para desplegar en entornos productivos. Todos los demás branches son no releaseables y significa que las versiones que se creen a partir de ellos sólo podremos desplegarlas en ambientes de test / beta.

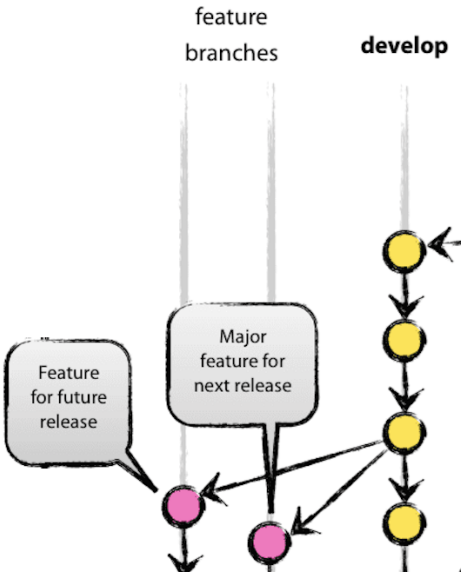
El flujo que nos plantea Git Flow es el siguiente:



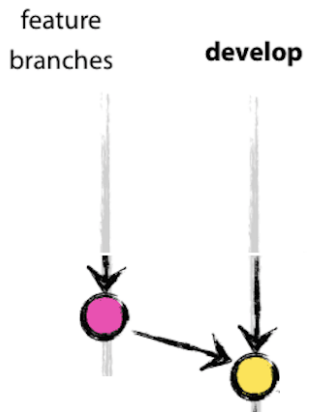
En Mercado Libre adoptamos el mismo flujo tal cual se describe en el diagrama, con algunas salvedades. Los branches revert-.*. enhancement/.*, fix/.* y bugfix/.* son sinónimos del branch feature/.* y los creamos porque en muchas ocasiones las epics / stories que vamos a trabajar durante los sprints no están orientadas a trabajar sobre nuevas funcionalidades, sino que en la corrección de errores (bugfixes) o mejoras técnicas (enhancements) y por ende, al agregarlos, intentamos que el flujo sea más semántico y también poder definir bien el contenido de cada versión.

Flujo paso a paso

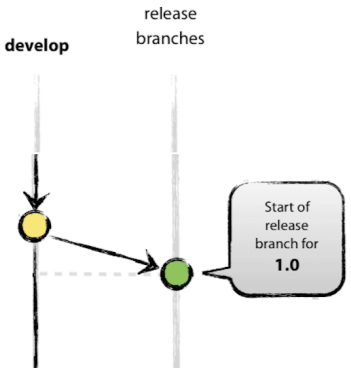
- **Paso 1:** A partir del branch develop creamos el/los branche/s feature/.* o sus análogos y comenzamos a trabajar en los cambios que queremos introducir.



- **Paso 2:** Una vez que terminamos de desarrollar los cambios, los testeamos y aseguramos que funcionan como deseamos, es momento de mergearlos o unirlos a la rama develop nuevamente. Para eso abrimos un pull request cuyo head branch es nuestro feature/.* y el base branch es develop.



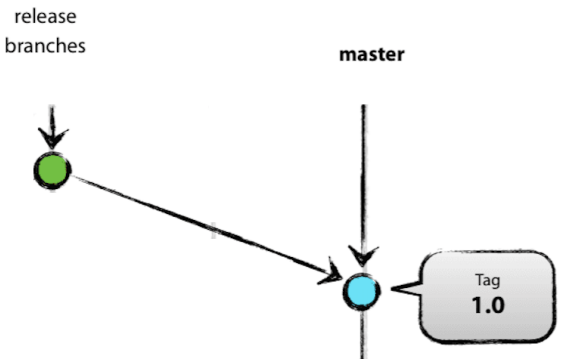
- **Paso 3:** Cuando queremos crear una versión para desplegar en producción, creamos un branch release/.* a partir develop para freezar los cambios y así evitar que se introduzcan nuevos no deseados en la versión que vamos a crear. Una vez creado el branch release/.*, es recomendable que generemos, con el comando “fury create-version”, una versión para poder desplegarla en un scope de test y realizarle todas las pruebas que creamos necesarias, en pos de asegurarnos de que el cambio que vamos a desplegar a producción es 100% correcto. En caso de encontrar errores, vamos a corregirlos en un branch fix/.* y creando un pull request, cuyo head branch sea ese fix/.* y el base branch sea ese release/.*.



- **Paso 4:** Cuando estemos seguros de que queremos desplegar en producción la release, abrimos un pull request cuyo head branch sea nuestro release/.* y el base branch sea master. Esto generará de manera automática el creado de una versión de tipo release candidate, que puede ser desplegada tanto en scopes productivos como de test.

Una vez abierto el pull request cuyo head branch sea release/.* y el base branch sea master, puede producirse una aprobación automática de la revisión de código según el caso de uso. Esto es gracias a que Gitflow garantiza para los escenarios donde sea aprobada, que todos los cambios puestos hasta ese momento, fueron admitidos en pull requests anteriores.

- **Paso 5:** Al realizar el merge de ese pull request, va a generarse automáticamente la versión que deberá ser desplegada en producción.



Paralelamente se abrirá, de forma automática, un nuevo pull request con esos cambios hacia el branch develop y hacia cualquier branch release/.* que se encuentre abierto. Esto se realiza con el fin de propagar los cambios realizados en el release, que acaba de ser mergeado, hacia la línea de desarrollo actual. Ese pull request se conoce con el nombre de backport.

Flujo de emergencia o hotfix

Cuando encontramos un error en una versión que está próxima a ser desplegada, o en la que ya está en un entorno productivo y tenemos que realizar una nueva versión de emergencia para arreglarlo, de manera rápida, existe un flujo que contempla este caso y nos permite 'saltar' temporalmente los controles de calidad de Release Process en pos de solucionar rápido el problema y luego inyectar calidad en el proceso.

El flujo consiste en:

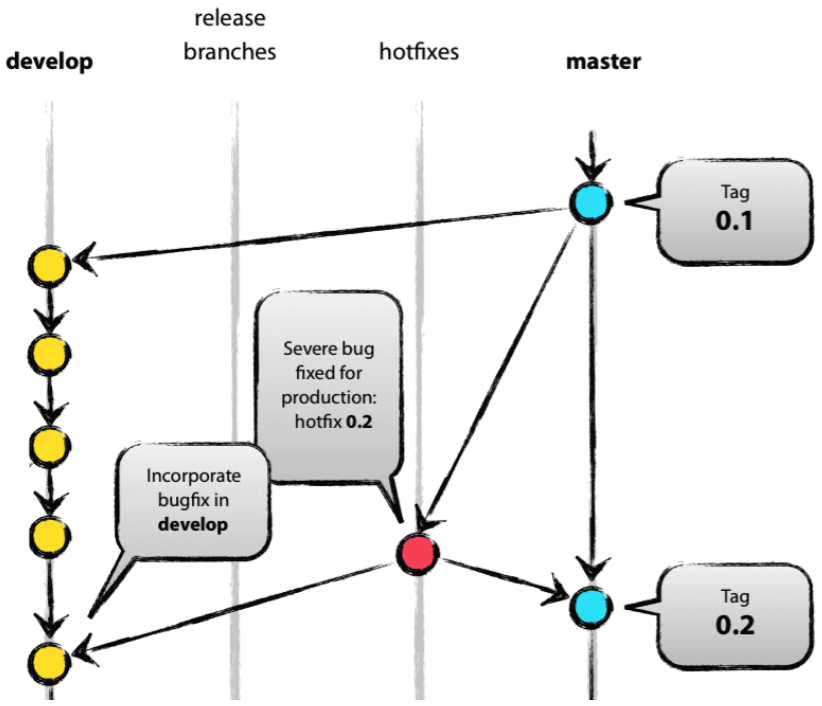
1. Creamos un branch hotfix/.* a partir del código del branch master (si el error fue encontrado en producción).

2. Escribimos los cambios en ese branch.

3. Armamos un pull request cuyo head branch será nuestro hotfix/.* y el base branch será master.

4. Si todos los chequeos de calidad fueron exitosos, se creará una versión automáticamente. En caso contrario, puede crearse una versión manualmente mediante el comando fury create-version

Una vez mergeado el pull request, se abrirá automáticamente un nuevo pull request con esos cambios hacia el branch develop y hacia cualquier branch release/.* que se encuentre abierto. Esto se realiza con el fin de propagar el arreglo del error a la línea de desarrollo actual y solucionarlo de raíz. Ese pull request se conoce con el nombre de backport.



Anexo 10: Reuniones realizadas

Reuniones y comunicaciones con el cliente

Minuta de comunicación de Junio 2021 - TEMÁTICA: Pruebas de concepto

Propuesta de identificación de perfiles de usuario

Para validar la idea y no invertir demasiado tiempo, elaboramos una prueba de concepto con una porción reducida de datos.

En base a un subset de 2 millones de compras, seleccionamos las filas que cumplen las siguientes condiciones:

- Compras realizadas a nivel L3 en MLB en una ventana de tiempo de 30 días.
- Usuarios que realizaron más de 10 compras en esa ventana de tiempo.

Algunas puntualizaciones:

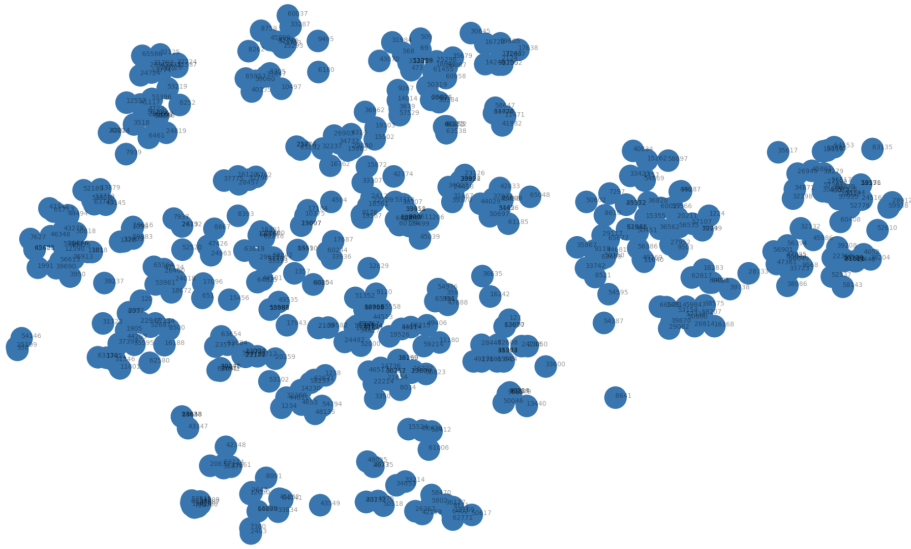
- Para seleccionar los datos, nos basamos en lo conversado anteriormente
- Esta solución es ampliamente escalable, utilizando las features de fury
- Nos concentramos en generar la data de perfiles y recomendaciones, antes de la presentación de la misma en un front.

Yendo un poco más a detalle teórico y técnico, la generación de la data de usuarios similares se generó de la siguiente manera:

- Tomamos los usuarios que forman parte de la muestra y los cruzamos con la/las categorías en las que han comprado, armando con esa info una matriz.
- Calculamos la similaridad entre los distintos usuarios: en base a las categorías en las que compró cada usuario, aplicando conceptos como distancia euclidiana y similitud del coseno (collaborative filtering).
- Una vez que tenemos los usuarios similares, aplicamos un algoritmo de clustering y obtenemos más información. Output: archivo con clusters de usuarios similares.

- En base a las compras realizadas, obtenemos las categorías que compran los usuarios similares, por lo que podemos recomendarle lo que otros hayan comprado. Output: archivo con categorías a recomendar por usuario.

Adjuntamos una captura de una prueba que hicimos con estos datos:



No se aprecia mucho el detalle de qué usuarios son, pero estuvimos analizando los grupos y las recomendaciones hacen sentido.

¿Cómo creamos valor para Mercado Libre con esta solución?

- Entendemos el comportamiento de los usuarios, al analizar similitudes entre ellos que no pueden ser identificadas a simple vista, por la escala del problema.

- Con este proceso, ofrecemos recomendaciones de ítems en función de los intereses de los usuarios que comparten el perfil, pudiendo cumplir con la idea original de ofrecer recomendaciones de ítems no necesariamente estrechamente relacionados (por ejemplo: a un tipo que compra todo cosas de gamer, podríamos ofrecerle galletitas dulces o cerveza si todos los usuarios similares están interesados en ellas). Todo esto en contraposición con las recomendaciones actuales de Mercado Libre que se hacen en base a ítems estrechamente relacionados a los visitados o adquiridos.

Reunión de seguimiento con cliente - Julio 2021

Agenda y minuta de la reunión

- **Validación de la forma de presentación de los datos de perfiles:** el cliente los prefiere en formato tabular.
- **Propuesta de conocer los usuarios asociados a un perfil:** el cliente parece interesado porque pueden surgir nuevos insights a partir de esta información.
- **Definición de métrica de éxito del proyecto:** se define que la métrica más importante será la de frecuencia de compra y que se evaluará a largo plazo.

Informe de avance Release 1 - Agosto 2021

E-Profiler - informe de avance y demo

Pablo Larralde <pablo.larralde@mercadolibre.com> dom, 29 ago 14:28

para Rafael, Rodrigo

Buenas tardes Rafa, ¿cómo estás?

Hace ya unos días tenemos en producción la primera fase del e-Profiler!

En esta oportunidad disponibilizamos un frontend en el que vas a poder acceder a un listado de perfiles de usuario, identificados a partir de compras de usuarios del segmento de usuarios que ya habíamos acordado. (Users de MLB que hayan realizado más de 10 compras distintas durante 2020, a nivel categorías L3).

Estos perfiles surgen, como ya comentamos, de aplicar algoritmos de clasificación y aprendizaje no supervisado sobre los datos obtenidos de Meli.

Para cada uno de ellos vas a disponer de la opción "View Users" donde podrás tener un detalle de los usuarios que componen cada perfil (id, nickname (por ahora genérico) y acceso a la info extra del usuario que proviene de la API de users).

El frontend quedó disponible en la URL: https://eprofiler.adminml.com, ya te dimos acceso para que puedas probarlo y darnos feedback.

Estamos muy contentos de poder haber publicado este release y de que ya puedas ir metiendo mano.

Nos ponemos a disposición para aclarar dudas o comentarte más detalles sobre este release.

Seguimos laburando para agregarle la parte de recomendaciones y envío a usuarios.

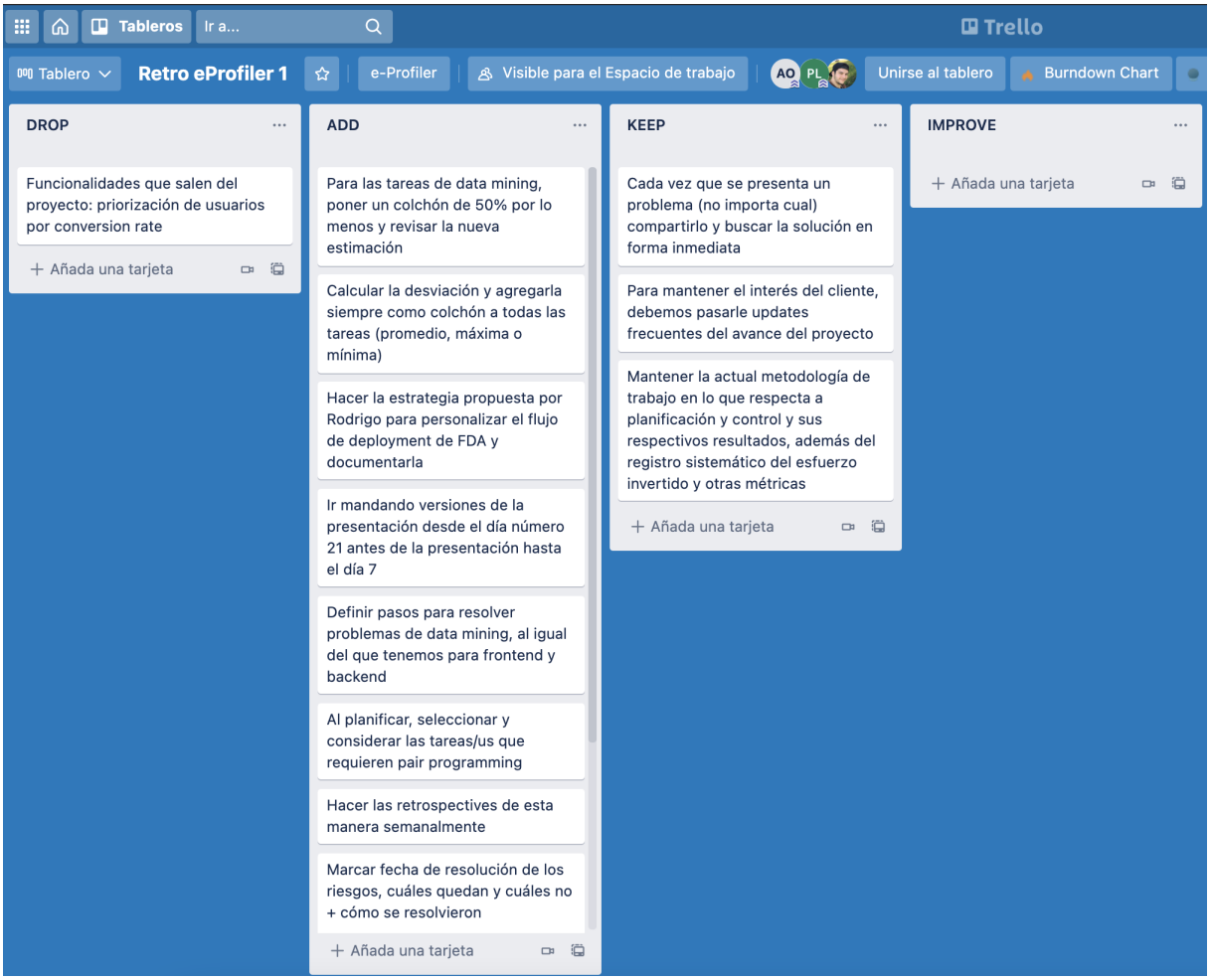
Informe de avance Release 2 - Setiembre 2021

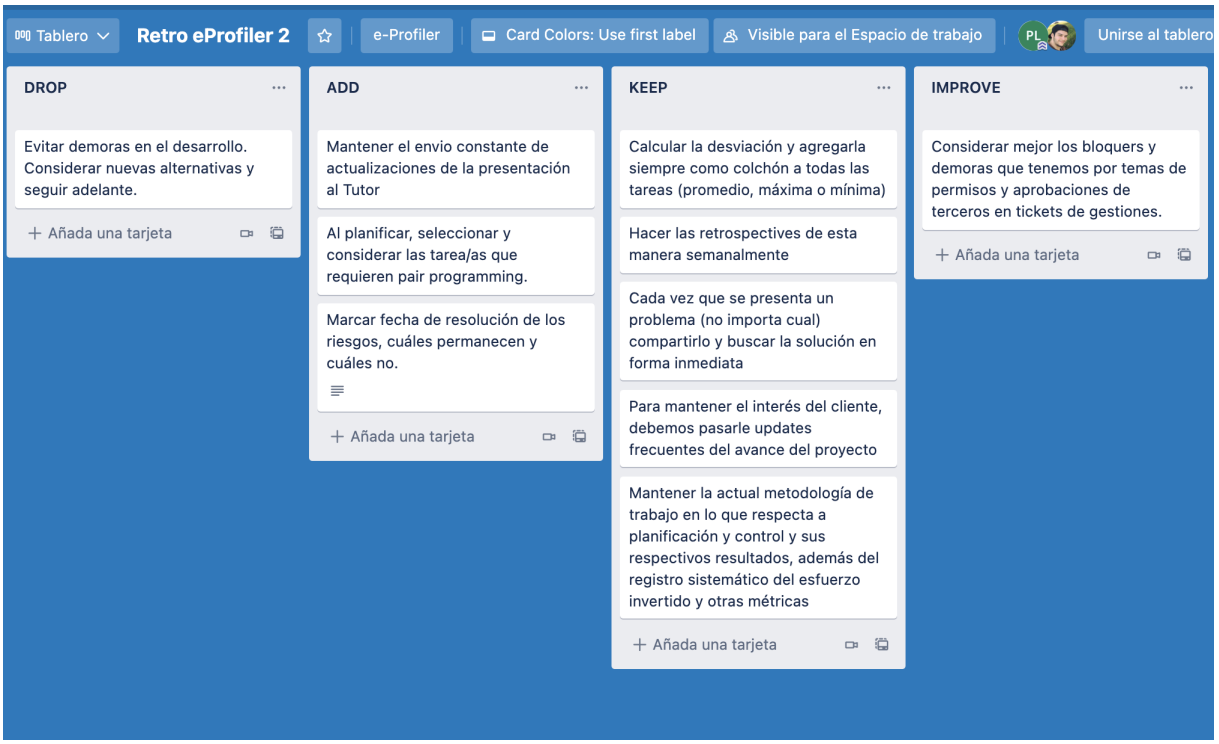
En esta oportunidad presentamos el producto al cliente en una videollamada y registramos la siguiente minuta:

- El producto e-Profiler cumple lo pactado, desde el punto de vista de los requerimientos del cliente.
- El cliente manifiesta que el equipo ha desarrollado el producto en el marco de un proceso serio y ordenado, lo cual le parece un gran diferencial.
- Queda pendiente la revisión de la métrica de frecuencia de compra para cuando el sistema quede al menos 6 meses en producción.
- Discutimos los pasos a seguir luego de la instancia de entrega académica y puesta en producción, como por ejemplo integrarnos con más equipos de la compañía.

Reuniones de retrospective del equipo

En esta sección, adjuntamos tableros de las reuniones de retrospective realizadas con el equipo y el tutor del proyecto. Aquí se refleja lo mencionado de la utilización de la metodología DAKI (Drop Add Keep Improve).



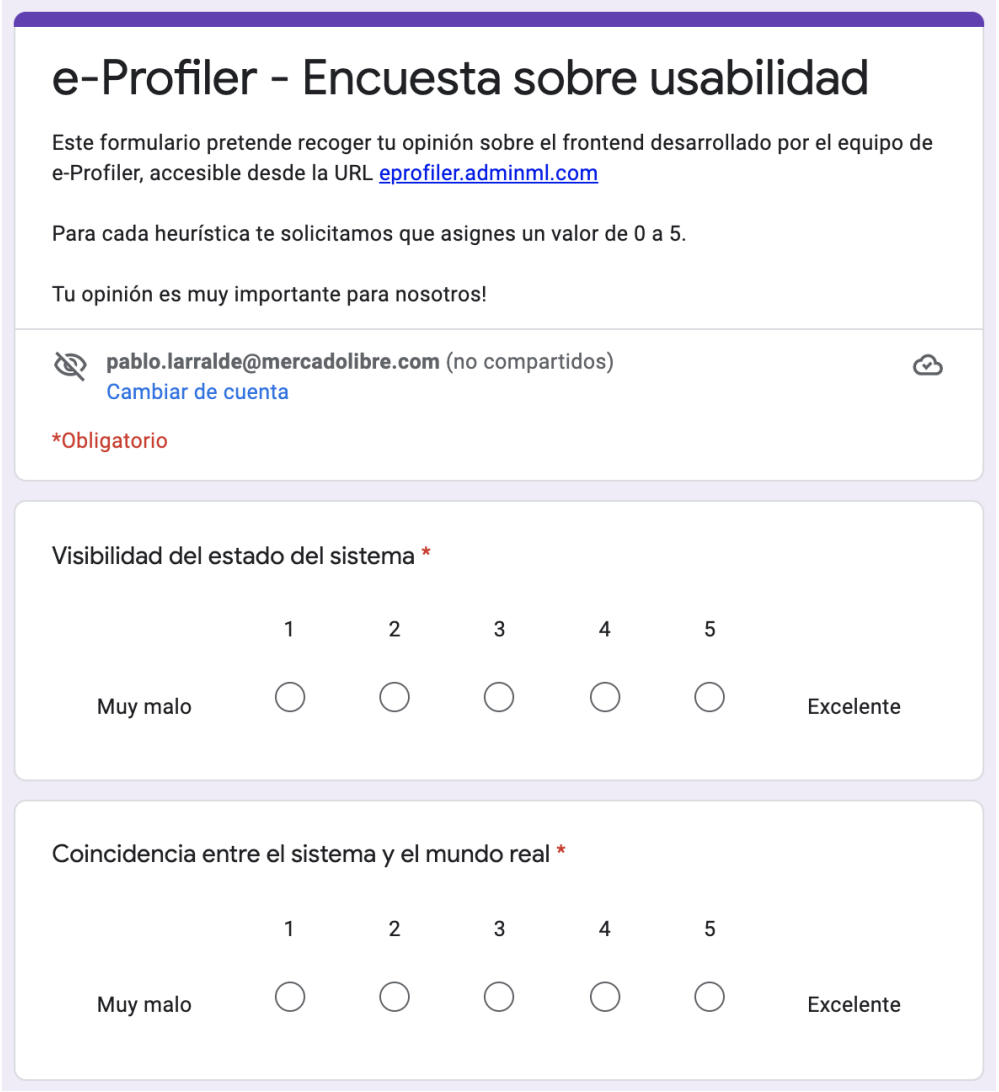

Tablero
Retro eProfiler 2
e-Profiler
Card Colors: Use first label
Visible para el Espacio de trabajo
Unirse al tablero
DROP
Evitar demoras en el desarrollo. Considerar nuevas alternativas y seguir adelante.
Añada una tarjeta
ADD
Mantener el envio constante de actualizaciones de la presentación al Tutor
Al planificar, seleccionar y considerar las tarea/as que requieren pair programming.
Marcar fecha de resolución de los riesgos, cuáles permanecen y cuáles no.
Añada una tarjeta
KEEP
Calcular la desviación y agregarla siempre como colchón a todas las tareas (promedio, máxima o mínima)
Hacer las retrospectives de esta manera semanalmente
Cada vez que se presenta un problema (no importa cual) compartirlo y buscar la solución en forma inmediata
Para mantener el interés del cliente, debemos pasarle updates frecuentes del avance del proyecto
Mantener la actual metodología de trabajo en lo que respecta a planificación y control y sus respectivos resultados, además del registro sistemático del esfuerzo invertido y otras métricas
Añada una tarjeta
IMPROVE
Considerar mejor los bloquers y demoras que tenemos por temas de permisos y aprobaciones de terceros en tickets de gestiones.
Añada una tarjeta


Tablero
Retro eProfiler 3
e-Profiler
Visible para el Espacio de trabajo
Unirse al tablero
Burndown Chart
DROP
Añada una tarjeta
ADD
Proponer al Product Owner utilizar la nueva herramienta de envío de recomendaciones en lugar de reinventar la rueda
Sintetizar la presentación para el tiempo de la defensa final
Explicar mejor la herramienta Fury en la presentación de la defensa final
Considerar demoras en los pedidos de soporte y aprobación de tickets de permisos
Añada una tarjeta
KEEP
Mantener estrategia de presentación de revisión 3 para la defensa final, ajustando la presentación con el feedback recibido
Mantener formas alternativas de comunicación con el Product Owner para no retrasar el feedback del proyecto y mantenerlo al tanto e interesado
Añada una tarjeta
IMPROVE
Añada una tarjeta

Anexo 11: Encuestas sobre el producto

Encuestas sobre usabilidad del frontend

En este anexo presentamos un ejemplo de las encuestas realizadas a distintos colegas de la empresa Mercado Libre. Las encuestas fueron implementadas con la herramienta de Google Forms (utilizada como forma estándar en la compañía para estas cuestiones). Las encuestas fueron distribuidas entre personas con perfil técnico y de negocio.



e-Profiler - Encuesta sobre usabilidad

Este formulario pretende recoger tu opinión sobre el frontend desarrollado por el equipo de e-Profiler, accesible desde la URL eprofiler.adminml.com

Para cada heurística te solicitamos que asignes un valor de 0 a 5.

Tu opinión es muy importante para nosotros!

pablo.larralde@mercadolibre.com (no compartidos)
Cambiar de cuenta

*Obligatorio

Visibilidad del estado del sistema *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Coincidencia entre el sistema y el mundo real *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Control y libertad del usuario *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Prevención de errores *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Flexibilidad y eficiencia de uso *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Mostrar en lugar de recordar *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Diseño estético y minimalista *

	1	2	3	4	5	
Muy malo	○	○	○	○	○	Excelente

Ayudar a reconocer, diagnosticar y recuperarse de errores *

	1	2	3	4	5	
Muy malo	◯	◯	◯	◯	◯	Excelente

Ayuda y documentación *

	1	2	3	4	5	
Muy malo	◯	◯	◯	◯	◯	Excelente

Enviar

Borrar formulario

Relevamiento de la satisfacción del cliente

En cuanto a este punto, por el momento realizamos un pedido de feedback oral al cliente, quien nos mencionó su calificación sobre el proyecto.

Los comentarios que surgieron de estas reuniones de review estuvieron orientados a que el producto cumplía con los requerimientos que nos solicitó. Nuestro cliente mencionó que además el proyecto fue realizado de una forma ordenada, teniendo en cuenta los riesgos, métricas de calidad y un proceso de ingeniería orientado por el tutor académico, por lo que satisface ampliamente.

El único punto que quedó pendiente de calificación fue la performance del proyecto en cuanto a la métrica fundamental del mismo, la frecuencia de compra. La calificación completa del cliente la tendremos cuando el proyecto esté en producción y puedan verse los resultados.

Anexo 12: Distribución de transacciones de Mercado Libre por país en el año 2020

A continuación presentamos información de cómo se distribuyeron las transacciones de ventas por país en el año 2020. Esto fue uno de los grandes sustentos que motivaron la elección del público objetivo que el cliente realizó al definir el alcance.

País	Millones de ventas en 2020
Brasil	1.461
Argentina	456
México	275
Otros	103