Diseño e Implementación de un Proceso de Recomendación para Plataforma de Bienestar Financiero

Carlos Daniel Montoya Hurtado

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesores
Luis Hernando Silva Flórez, Magíster (MSc) en Ingeniería
Juan José Martínez Velásquez, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2022

TABLA DE CONTENIDO

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

AWS	Amazon Web Services
CQRS	Command Query Responsability Segregation
DTO	Data Transfer Object
QA	Quality Assurance

GLOSARIO

Data Lake Repositorio de almacenamiento centralizado, fácilmente accesible, para grandes volúmenes de datos estructurados y no estructurados.

Endpoint Interfaz expuesta por un comunicante o un canal de comunicación, que puede ser utilizada a través de una llamada a un procedimiento remoto.

ETL Del inglés *Extract, transform and load*, es un proceso que permite mover datos desde múltiples fuentes, transformarlos y cargarlos en otro sistema de almacenamiento para que sean aprovechables.

Evento Unidad de información que encapsula el resultado un cambio en el sistema, como la creación o modificación de datos. Usado para notificar el resultado de los cambios realizados a otros interesados.

Experiencia Flujos de elementos, principalmente interfaces graficas de usuario, con los que el usuario puede interactuar con los servicios que provee la plataforma, los cuales, mediante implementaciones propias o integraciones con terceros, como cuestionarios o consultas a calificadoras de riesgo, recolectan datos que permiten describir el estado financiero de la persona.

Quality gate Función que utiliza SonarQube para asegurar el cumplimiento de la Política de Calidad en una organización.

Release Plan O plan de proyecto, es un conjunto de historias de usuario (normalmente épicas) agrupadas por *releases* o versiones del producto que se ponen a disposición de los usuarios incrementando el valor para estos respecto de la anterior.

Retrospectiva	Espacio en el que se planean formas de incrementar la calidad y efectividad. Se inspecciona como fue el último sprint respecto a los individuos, las interacciones, los procesos, herramientas, y la definición de finalizado. Se discute que fue bien durante el Sprint, que problemas se encontraron, y como esos problemas fueron (o no fueron) solucionados.
Service mesh	Infraestructura de software dedicada a manejar la comunicación entre microservicios. Proporciona y permite aplicaciones basadas en contenedores y microservicios, los cuales se integran directamente desde el interior del clúster.
Sprint	Eventos de duración fija de un mes o menos para crear consistencia. Un nuevo Sprint inicia inmediatamente después de la conclusión del Sprint previo. Todo el trabajo necesario para alcanzar el objetivo del producto, incluyendo la Planeación del Sprint, Dailies, Revisión del Sprint y Retrospectiva del Sprint, suceden en los Sprints.
Stream	Conjunto de señales digitales usadas para transmitir diferentes tipos de contenido de manera continua, como eventos.
Visual Story Mapping	Técnica comúnmente utilizada para representar de forma visual necesidades o una idea de producto, brindando como resultado versiones del producto y plan de entregas, ayudando a tener un panorama más claro de lo que se va a construir.

RESUMEN

Las recomendaciones de servicios y consejos adecuados pueden ser un medio para mejorar el bienestar financiero de una persona. La personalización de éstas es realizada mediante modelos de recomendación a partir de los datos de cada usuario. En la plataforma existen múltiples fuentes de datos heterogéneos distribuidas en diferentes componentes de transmisión y almacenamiento, y diversos modelos de recomendación que son usados por muchos clientes y procesos mediante peticiones acopladas a cada uno de ellos. A través de abstracciones se diseñó un proceso de recomendación que permite a cualquier cliente o proceso dentro de la plataforma interactuar con múltiples modelos de recomendación de manera transparente, independientemente de los algoritmos con los que se hayan implementado. Las recomendaciones, y las características a partir de las que se calculan, pueden ser generadas en procesamientos en línea, a partir de consultas síncronas; casi en tiempo real, mediante eventos enviados en streams de datos; y en lote, recurriendo a ETLs; siendo capaces de explotar las diversas fuentes de datos que hacen parte de la plataforma. El componente central de la implementación realizada es un microservicio encargado de exponer a los demás clientes de la plataforma las funcionalidades para calcular y consultar recomendaciones; de gestionar la extracción y almacenamiento de las características a partir de los datos de los usuarios; y de orquestar los modelos de recomendación adecuados para cada contexto. Los componentes resultantes se diagramaron utilizando el modelo C4, centralizando la documentación y unificándola bajo un mismo estándar intuitivo y adaptable a diferentes públicos.

Palabras clave — **Sistemas de recomendación, Arquitectura de software, Procesamiento en línea, Procesamiento en lote, Microservicios**

ABSTRACT

Services and tips recommendations can be a way to improve a person's financial well-being. The personalization of these recommendations is achieved through recommendation models using each user's data. There are multiple heterogeneous data sources on the platform, distributed on different streaming and storing components, and various recommendation models that are used by many clients and processes via requests that are coupled to each one of them. Through abstractions, we designed a recommendation process that allows to any client or process within the platform to interact with multiple recommendation models in a seamless manner, regardless of the algorithms in wich thay have been implemented. Recommendations, and the characteristics from which they are calculated, can be generated in online processing; on synchronous requests; in near-real-time, from events sent in data streams; and in batch, resorting to ETLs; being able to exploit the various data sources that are part of the platform. The central component of the implementation carried out is a microservice in charge of exposing to the other clients in the platform the functionalities to calculate and consult recommendations, to manage the extraction and storage of features from users data; and to orchestrate the appropiate recommendation models for each context. The resulting components were diagrammed using the C4 model, centralizing the documentation and unifying it under the same intuitive, adaptable to different audiences standard.

Keywords — **Recommendation system, software architecture, online processing, batch processing, microservices**

I. INTRODUCCIÓN

Un sistema de recomendación es el componente responsable de intentar determinar cuál producto es más probable que adquiera un usuario en un momento dado [1]. A través de las recomendaciones, la plataforma busca mejorar el bienestar financiero de las personas, ofreciendo consejos y servicios financieros de acuerdo con el conocimiento que se tiene del usuario, el cual puede abarcar datos sociodemográficos, historial crediticio, autopercepción de hábitos financieros, entre otros. Estos datos se obtienen a medida que el usuario navega a través de experiencias.

Las experiencias son flujos de elementos, principalmente interfaces gráficas, con los que el usuario puede interactuar con los servicios que provee la plataforma, los cuales, mediante implementaciones propias o integraciones con terceros, como cuestionarios o consultas a calificadoras de riesgo, recolectan datos que permiten describir el estado financiero de la persona. Los datos obtenidos se transmiten en tiempo real como eventos a través de streams, y son modelados en un dominio, antes de ser persistidos en bases de datos relacionales y en un data lake. Estos dominios hacen parte de contextos delimitados en los que se define el entendimiento organizacional de los datos y las funcionalidades de la plataforma desarrolladas alrededor de ellos. Usualmente, una experiencia se corresponde con un contexto delimitado.

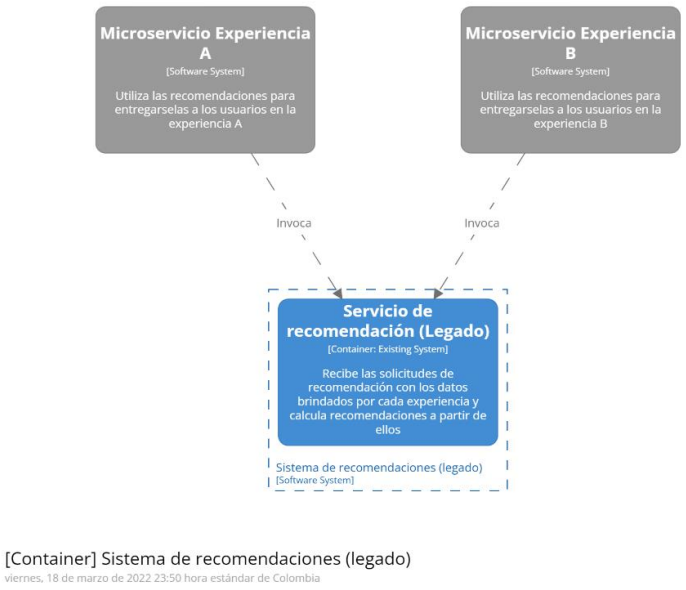


Fig. 1: Sistema de recomendaciones legado

La primera aproximación de la plataforma en la generación y entrega de recomendaciones surge específicamente en una de las experiencias, para la cual se implementó un modelo de segmentación que recibe una petición HTTP síncrona, en la 6 que el cliente (en este caso la experiencia) envía el conjunto de datos que tiene a su disposición. El modelo extrae algunos indicadores básicos de los datos enviados en la petición y, con ellos, realiza el proceso de segmentación y calcula las recomendaciones. Esta implementación inicial está altamente acoplada a la experiencia pionera en el uso de las recomendaciones, y a medida que surgieron nuevas experiencias en la plataforma, y otras ya existentes adoptaron estas en sus flujos, se obligó a reestructurar el recomendador para que fuera capaz de soportar las solicitudes de múltiples experiencias (fig. 1), donde cada una de ellas tiene sus particularidades, como los datos enviados, las técnicas necesarias para calcular las recomendaciones o las reglas específicas de cada experiencia para ser entregadas al usuario. Además, en el proyecto se finalizó una consultoría, en la cual uno de los entregables fue un nuevo modelo de recomendación basado en grafos, que para funcionar correctamente depende de unas librerías específicas que no podían ser incluidas en el sistema de recomendación original. Se hizo evidente que en esta aproximación del sistema de recomendación se presentan varios inconvenientes:

- **No se utilizan todos los datos del usuario de los diferentes dominios:** Debido a que cada experiencia envía los datos que tiene en su contexto delimitado, cuando se calculan las recomendaciones no se tiene el espectro completo de los datos de un usuario, por lo que al calcular las recomendaciones de un usuario en la experiencia A, no se podía contar con un atributo o indicador de la persona obtenido en la experiencia B.
- **No esta estandarizada la introducción de nuevos recomendadores:** Los recomendadores que se implementaron para cada experiencia fueron realizados a demanda, resolviendo la necesidad específica en el momento, pero sin implementar mecanismos que permitan integrar de manera sistemática nuevos recomendadores.
- **El proceso no es reactivo al flujo de información que sucede en la plataforma:** En la plataforma, cada cambio en el estado del sistema genera un evento en el que viaja mucha información. Esta información viaja en tiempo real a través de streams

de datos, los cuales no están siendo utilizados en esta aproximación ya que sólo se tienen en cuenta los datos que se reciben en la solicitud de recomendaciones

El sistema inicial empezó a crecer en complejidad. La falta de una estrategia concreta para lograr integraciones entre los diferentes servicios de la plataforma y el recomendador inicial, la nula transparencia para interactuar con los sistemas de recomendación, además de no seguir los patrones de arquitectura empresarial definidos en la organización, y la ausencia de una implementación de pruebas y un proceso de aseguramiento de la calidad definido supuso que se planteara la necesidad de crear un proceso estándar de recomendación para toda la plataforma, que se apoyara en un microservicio responsable de articular los diferentes componentes que hacen parte del proceso de recomendaciones, además de habilitar el uso de la información histórica, transaccional y en tiempo real disponible en la plataforma. El diseño de este nuevo proceso es realizado dentro de la implementación propia de la plataforma del marco de trabajo SCRUM, descrito en la sección IV.

Así, se espera que las recomendaciones entregadas a los usuarios sean más acertadas, ya que los sistemas de recomendación tendrán a su disposición toda la información extraída de un usuario desde los diferentes dominios y fuentes de datos de la plataforma, se podrán ejecutar modelos de recomendación más complejos de manera transparente para el resto de la plataforma, será más eficiente la introducción de nuevos sistemas de recomendación y el esfuerzo de integración de las experiencias hacia los sistemas de recomendación ser a disminuido drásticamente, y se mejorar a la efectividad de las recomendaciones entregadas, al habilitar mecanismos de preprocesado y post-procesado que interactúan con el estado no solo del usuario, sino también de toda la plataforma como tal.

II. OBJETIVOS

A. Objetivo general

Diseñar e implementar un proceso de recomendación que permita, a partir de los datos consolidados de un usuario, calcular recomendaciones síncrona o asíncronamente, habilitando mecanismos de comunicación al resto de microservicios y procesos de la plataforma de una forma estándar.

B. Objetivos específicos

- Diseñar la arquitectura para el proceso de recomendación siguiendo el modelo estándar que implementa la plataforma.
- Definir las entidades y funcionalidades que representan el entendimiento del proceso de recomendación.
- Implementar los componentes necesarios para plasmar el diseño del proceso de recomendación.

III. MARCO TEÓRICO

La plataforma está compuesta, a alto nivel, por una aplicación m ovil nativa en IOS y Android, una página web y un backend, estos dos últimos operando en la nube de Amazon Web Services (AWS) [2], y una capa de persistencia en Google Cloud Platform. El backend implementa el patrón de arquitectura de microservicios, el cual es una aproximación para desarrollar una aplicación completa como un conjunto de pequeños servicios, cada uno corriendo su propio proceso y comunicándose a través de mecanismos ligeros [3]. Cada microservicio corresponde a la implementación de un dominio modelado en un contexto delimitado [4], representado en la figura 2, lo que permite reducir la complejidad del modelado del dominio de todo el sistema completo al concentrarse en una parte específica del dominio y no tener que preocuparse de los detalles de implementación de otros subdominios.

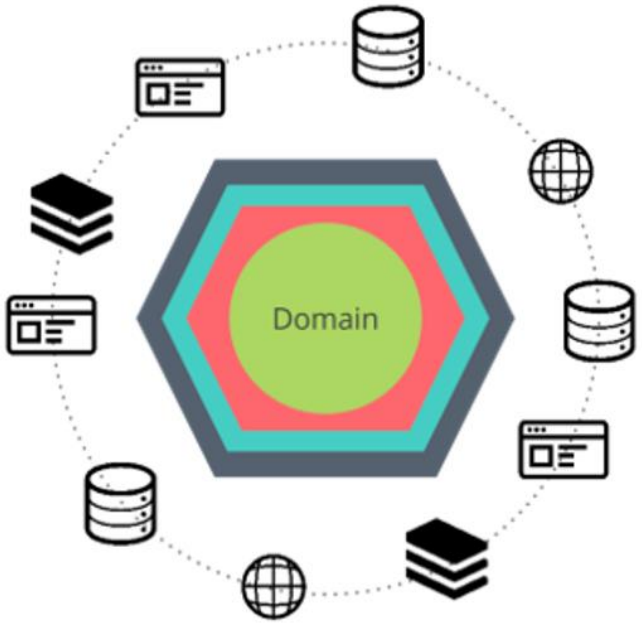
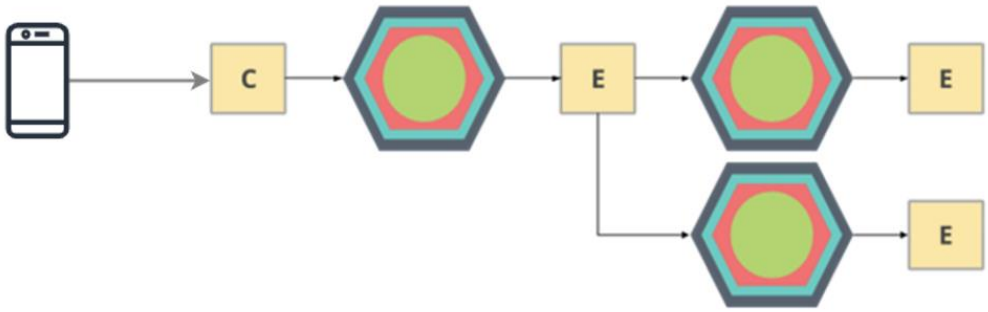


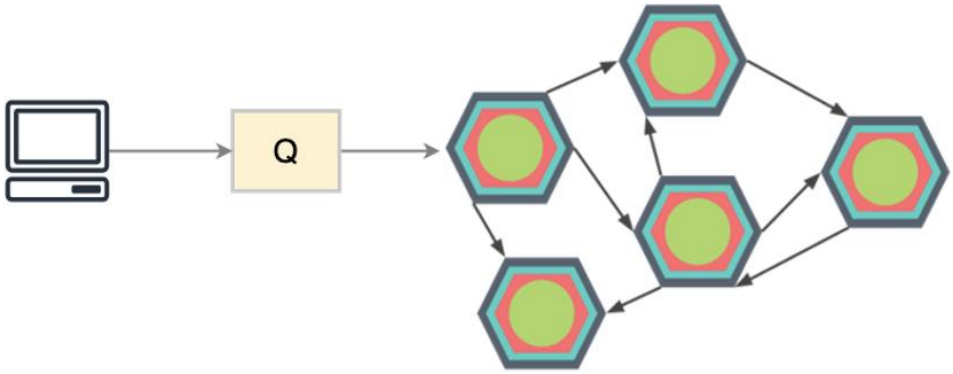
Fig. 2: Microservicio encapsulando un dominio en su contexto delimitado

Los microservicios dentro de la plataforma se comunican tanto síncrona como asíncronamente, la primera se da mediante consultas e invocaciones de servicios HTTP (fig. 3b) y la segunda a través de la ejecución de comandos desencadenados por los eventos que se emiten y reciben entre microservicios (fig. 3a). Este mecanismo de comunicación se apoya en la

implementación del patrón Command Query Responsability Segregation (CQRS), el cual permite separar y escalar independientemente las cargas de lectura y escritura del microservicio [5], como se puede observar en la figura 3. Cada evento es el resultado de modificar el estado del sistema, como crear o actualizar una entidad de dominio [6], estos son generados y almacenados secuencialmente para tener una traza de la evolución del sistema a lo largo del tiempo [7]. Estos microservicios son los clientes principales de las recomendaciones, encargados de llevar las recomendaciones calculadas para un usuario a los diferentes flujos y experiencias definidas.



(a) La ejecución de un **comando** emite como resultado un evento que desencadena procesamientos en los microservicios suscritos a dicho evento.



(b) Al realizar una **consulta**, el microservicio recupera el estado de los microservicios necesarios para componer la respuesta.

Fig. 3: Patrón CQRS

Las recomendaciones pueden ser calculadas en diferentes modos de ejecución o momentos, como se detalla en [8]: pueden existir procesos de recomendación capaces de entregar recomendaciones en escenarios online, en los que se responde a las interacciones del usuario en

tiempo real, respondiendo rápidamente y usando los datos m as recientes adquiridos; también pueden ser calculadas offline, normalmente en procesos en lote que tienen mayor flexibilidad en la complejidad de los algoritmos y la cantidad de datos usables para el cálculo de las recomendaciones, pero con un tiempo de respuesta significativamente mayor; y pueden calcularse en escenarios nearline, en los que se busca un compromiso intermedio entre los dos modos anteriores, permitiendo cálculos rápidos o complejos sin el compromiso de entregarlos en tiempo real, siendo este un mecanismo de cálculo principalmente asíncrono.

Extraer información de las interacciones de los usuarios, calcular y entregar recomendaciones de una forma escalable tanto en tiempo real como asíncronamente significa un reto de diseño a nivel de arquitectura. [9] propone una variedad de arquitecturas de referencia para diferentes casos de uso, tales como crear mecanismos de cache para la entrega de predicciones, realizar procesos de disponibilización de datos en lote, o de extracción de características dinámicas en tiempo real, brindando diagramas de alto nivel que pueden ser adaptados según la necesidad.

Para que las características puedan ser usadas en la operación de la plataforma en tiempo real, se deben persistir en un sistema de almacenamiento apropiado. Los Feature Stores [10] son un concepto emergente en las arquitecturas organizacionales, permitiendo centralizar las características creadas a partir de múltiples fuentes de datos para ser usadas tanto para el entrenamiento de modelos, como por las aplicaciones que no desean calcular las características, sino utilizarlas en el momento en que se necesiten hacer predicciones.

[11] propone un modelo para la documentación y presentación de los diseños de arquitecturas de software a partir de abstracciones jerárquicas que permiten visualizar los sistemas desde diferentes puntos de vista, partiendo desde un alto nivel en el que se plasma el contexto del sistema; pasando por contenedores que representan artefactos que deben estar operando para que el sistema funcione; componentes que definen las funcionalidades que encapsula un contenedor; hasta llegar al nivel de código, diagramas de secuencias, diagramas de clases, entre otros aspectos, a nivel granular dentro del sistema. Este modelo brinda una notación unificada que facilita mantener la consistencia entre los diferentes diagramas que describen al sistema, de una manera intuitiva.

IV. METODOLOGÍA

El desarrollo de la plataforma se da dentro del marco de las metodologías agiles, adaptando el marco de trabajo SCRUM [12], mostrado en la figura 4, a las necesidades organizacionales. El equipo de trabajo se divide en dos niveles llamados frentes y células. Un frente se enfoca en un objetivo específico organizacional, y las células que hacen parte de ese frente realizan actividades que ayudan a la consecución del objetivo.

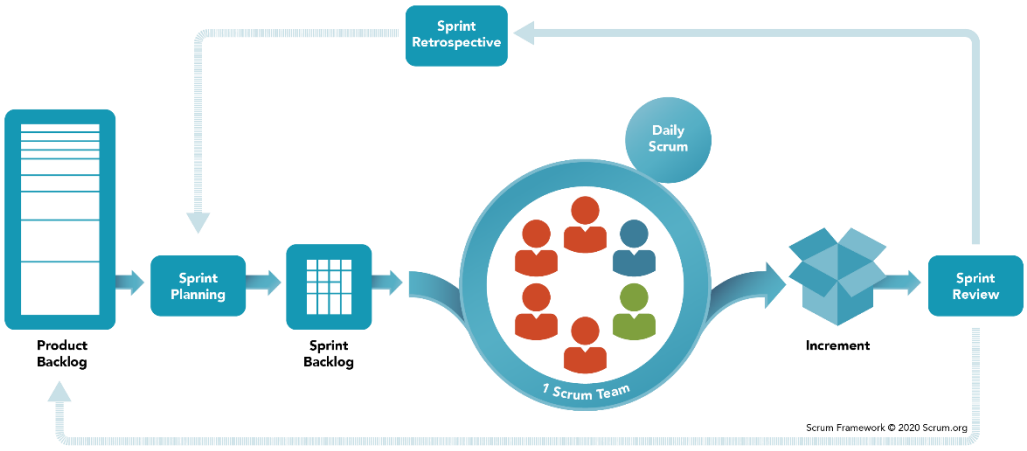


Fig. 4: SCRUM Framework

La ejecución de estas actividades se realiza en sprints de dos semanas de duración en las que se hace seguimiento diario a los avances en las reuniones de sincronización o dailies, al final del sprint se realiza una revisión de los resultados alcanzados, se hace una retrospectiva de lo que salió bien, qué puede mejorar y qué acciones se pueden realizar para materializar esa mejora y, finalmente, se planean y priorizan las actividades del próximo sprint. En el proyecto, estos espacios, conocidos en SCRUM como eventos o ceremonias, se realizan tanto a nivel de célula de trabajo como a nivel general del proyecto, con el fin de que los diferentes equipos estén sincronizados respecto al estado del producto

Se partió de un entendimiento de la necesidad a través de reuniones con diferentes interesados en incorporar las recomendaciones en la plataforma desde perspectivas tanto tecnológicas como de negocio. En las reuniones se cuestión o como se imaginaba la generación y

entrega de recomendaciones en la plataforma, que capacidades les gustaría que soportara el proceso de recomendación y qué funcionalidades les gustaría soportar a día de hoy y en un futuro. Con esto se buscó determinar el alcance inicial del proceso de recomendación, pero también, cuáles son los puntos en los que debe evolucionar el sistema, de tal manera que el diseño e implementación resultante de la primera versión tenga en cuenta la visión a futuro del proceso de recomendación y puedan ser extendidos fácilmente.

De estas sesiones se concluyó que se desea que el proceso de recomendación sea capaz de entregar las recomendaciones más relevantes a un usuario dado el conocimiento que se tiene de el en el momento. Es decir, se espera que, para los usuarios, independientemente de la cantidad de datos que se tengan de ellos, se puedan calcular y entregar recomendaciones, las cuales pueden ir evolucionando a medida que se adquieran mas datos del usuario. Basados en la necesidad, se definió un conjunto de requisitos funcionales y no funcionales. Este recoge las capacidades y atributos de calidad que se espera que tenga el proceso de recomendación y, además, sus futuras evoluciones soporten.

Basados en el entendimiento obtenido y los requisitos definidos, se realizó una búsqueda de referentes que se hubieran enfrentado a retos similares en el contexto de las recomendaciones, descritos en la sección III. Después de esto, se definió como objetivo de un sprint tener un diseño inicial que supliera las necesidades y cumpliera con los requisitos definidos. Los referentes identificados inspiraron fuertemente el diseño de la solución para el caso de uso de nuestra plataforma, y, además, expusieron consideraciones adicionales que resultaron de suma utilidad para obtener un diseño extensible. El diseño fue llevado, presentado y evaluado en el Comité de Arquitectura del proyecto, un espacio en el cual se reúnen desarrolladores, arquitectos, especialistas, entro otros roles técnicos, y se discuten problemas, retos o dificultades y se plantean soluciones a ellos. Luego de una retroalimentación por parte de los asistentes, se aceptó el diseño y se procedió a planear la implementación.

A partir del diseño, se plantearon las diferentes tareas que se consideraron necesarias para poder materializarlo. Estas tareas fueron organizadas a través de un Visual Story Mapping y segmentadas en un Release Plan. El alcance de la primera versión del proceso de recomendación fue definido para que se incorporaran los modelos de recomendación existentes y que, además, se integrara el modelo de recomendación basado en grafos, añadiendo unos pasos de post-procesamiento de las recomendaciones generadas por los modelos considerados de gran valor

para exponer correctamente las recomendaciones en las experiencias a los usuarios, tales como no entregar recomendación de un servicio que ya el usuario haya adquirido

Una vez trazado el plan, se inició la fase de implementación del primer hito en el release plan, realizada a lo largo de diferentes sprints. Cada funcionalidad implementada fue acompañada por sus respectivas pruebas unitarias y de integración. Después de haber implementado las funcionalidades esperadas en el primer release, se realizó una fase de integración, en la cual se modificaron los sistemas existentes para utilizar el proceso de recomendación, en detrimento del sistema de recomendaciones original. Finalizado el proceso de desarrollo, se llevaron a cabo un conjunto de exhaustivas pruebas en los diferentes componentes de la plataforma que fueron afectados por el cambio, para así garantizar una transición correcta al nuevo proceso de recomendación. Estas pruebas fueron realizadas en conjunto con el equipo de Quality Assurance (QA), y se abarcaron pruebas de integración, pruebas funcionales y pruebas de rendimiento, con las que se certificó que el desarrollo estaba listo para ser promovido a un ambiente productivo.

La implementación de los microservicios en la plataforma se realizó en el lenguaje de programación Java, apoyándose en el Play [13], el cual permite definir endpoints que exponen los comandos y las consultas definidas en el microservicio, soporta migraciones en el esquema de la base de datos usando evoluciones [14] y facilita la serialización y deserialización de los objetos transferidos a través de los diferentes protocolos de comunicación, entre otras capacidades. Además, en cada microservicio se utilizó la librería VAVR [15], esta brinda estructuras de datos y de control que facilitan la implementación del paradigma de programación funcional. Las pruebas unitarias se implementaron usando la librería JUnit [16] en conjunto con la librería Mockito [17], que permite crear Test doubles [18]. Para el aseguramiento de la calidad de los artefactos y de las pruebas, se realizó tanto un análisis estático como de cobertura de las pruebas implementadas, el resultado fue evaluado en la herramienta SonarCloud ante unos quality gates [19] que, en caso de no ser superados, impiden promover los artefactos a un ambiente productivo hasta ser corregidos. Cada microservicio sigue el principio de Base de datos por servicio [20], utilizando un motor de base de datos relacional para mantener el estado del microservicio.

Finalmente, se consolido una documentación en la cual se describe el origen del proceso de recomendación, su diseño e implementación, guías para que otros integrantes del proyecto puedan modificar, agregar e incorporar los sistemas de recomendación a los diferentes flujos en los que se considere necesario el uso de recomendaciones. En este punto cobro gran relevancia el

uso del modelo C4 [11], el cual es un modelo propuesto para diagramar arquitecturas de software, destacado en [21] por la simplicidad y facilidad con la que es capaz de comunicar los componentes que hacen parte de un sistema y las interacciones que se dan entre ellos.

V. RESULTADOS

Las recomendaciones son consideradas un elemento diferencial en la plataforma, siendo un medio para materializar los objetivos organizacionales y, como tal, se espera que puedan acompañar a los usuarios en diferentes escenarios, flujos y procesos, sea a través de las experiencias en la aplicación web o en la aplicación móvil, mediante correos, servicios de mensajería instantánea, notificaciones o incluso campañas de atracción. Asimismo, estas pueden ser calculadas mediante diferentes algoritmos, como técnicas de segmentación, representación de recorridos usando grafos, entrenando modelos de aprendizaje automático, usando filtrados colaborativos, entre muchas otras aproximaciones.

Para que la gran variedad de escenarios en los que se pueden calcular y entregar recomendaciones puedan coexistir de manera armoniosa se necesita un mecanismo estándar para que puedan interactuar. Para lograr diseñar este mecanismo, se definió un conjunto de abstracciones (fig. 5) a partir de las cuales se puede construir un lenguaje común y permiten componer flujos complejos que representan el proceso de recomendación deseado. Cada uno de los puntos en los que se puede entregar una recomendación es un contexto; las recomendaciones son calculadas por recomendadores a partir de las características de un usuario; y un orquestador se encarga de proporcionar a los recomendadores las características requeridas para poder realizar predicciones, y de entregar a cada contexto las recomendaciones calculadas por los recomendadores, funcionando, este, como un intermediario.

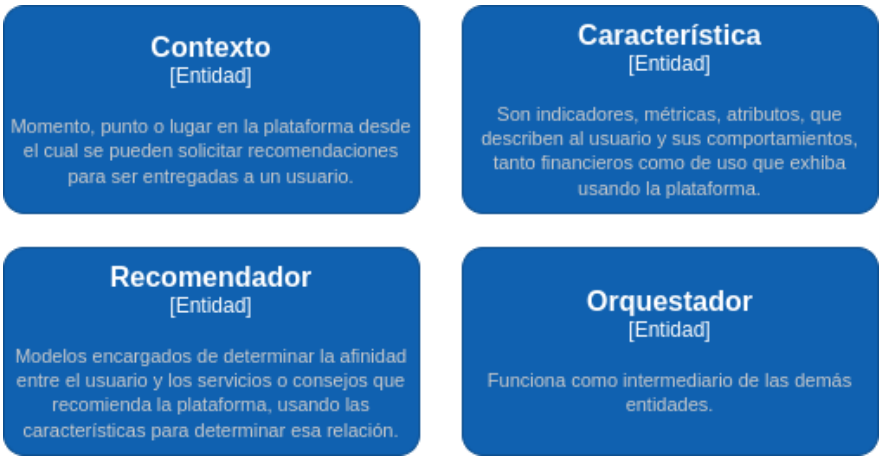


Fig. 5: Entidades abstraídas en el proceso de recomendación

Estas entidades en conjunto conforman el Sistema de Recomendación, este es el encargado de gestionar todo el proceso de cálculo y entrega de recomendaciones, exponiendo servicios que pueden ser usados por otros microservicios o procesos que conforman la plataforma para ser incorporados en experiencias o flujos particulares, o directamente ser utilizados desde las aplicaciones web y móvil, como se muestra en la figura 6. El sistema de recomendación se apoya en un Servicio de utilidades, que le sirve para complementar el proceso de entrega de recomendaciones, permitiendo, por ejemplo, filtrar servicios que los usuarios ya hayan contratado previamente. Además, el sistema interactúa directamente con el sistema de mensajería de la plataforma, leyendo y publicando eventos al stream general. De esta manera, el sistema de recomendación puede extraer información de todos los cambios que se den en el estado de la plataforma, y convertirlas en características usables para calcular recomendaciones más precisas.

Para la plataforma es de suma importancia poder aprovechar lo máximo que sea posible la atención del usuario en el momento en que el navega a través de las diferentes experiencias, por lo que el cálculo de recomendaciones de manera online toma especial relevancia para poder entregarle recomendaciones personalizadas a su situación financiera particular, y así este pueda percibir el conocimiento y valor que le puede aportar la plataforma a su bienestar financiero. Sin embargo, no deja de ser importante el conocimiento que se ha consolidado del usuario a partir de los datos adquiridos, por lo que si bien se busca entregar recomendaciones de manera online con los datos disponibles en el momento, también se desea poder aprovechar todo aquel conocimiento previo que se tenga, buscando recomendaciones mucho más adecuadas a cada usuario, haciendo que los modos de ejecución nearline y offline sean también piezas clave en el proceso de recomendación de la plataforma.

Los datos históricos resultantes de la interacción de los usuarios con la plataforma son almacenados en el data lake. Dado el gran volumen de los datos, y la naturaleza no transaccional del data lake, los datos son transformados de manera offline mediante ETLs especializadas que recorren los diferentes dominios de datos aplicando diversos cálculos y operaciones que resultan en características. Estas son persistidas en una base de datos de baja latencia que es usada como un repositorio de características (o Feature Store) para procesos tanto offline como online, permitiendo así que una vez las características sean calculadas, estén disponibles para que el microservicio recomendador pueda realizar operaciones en tiempo real, como se puede observar en la figura 7.

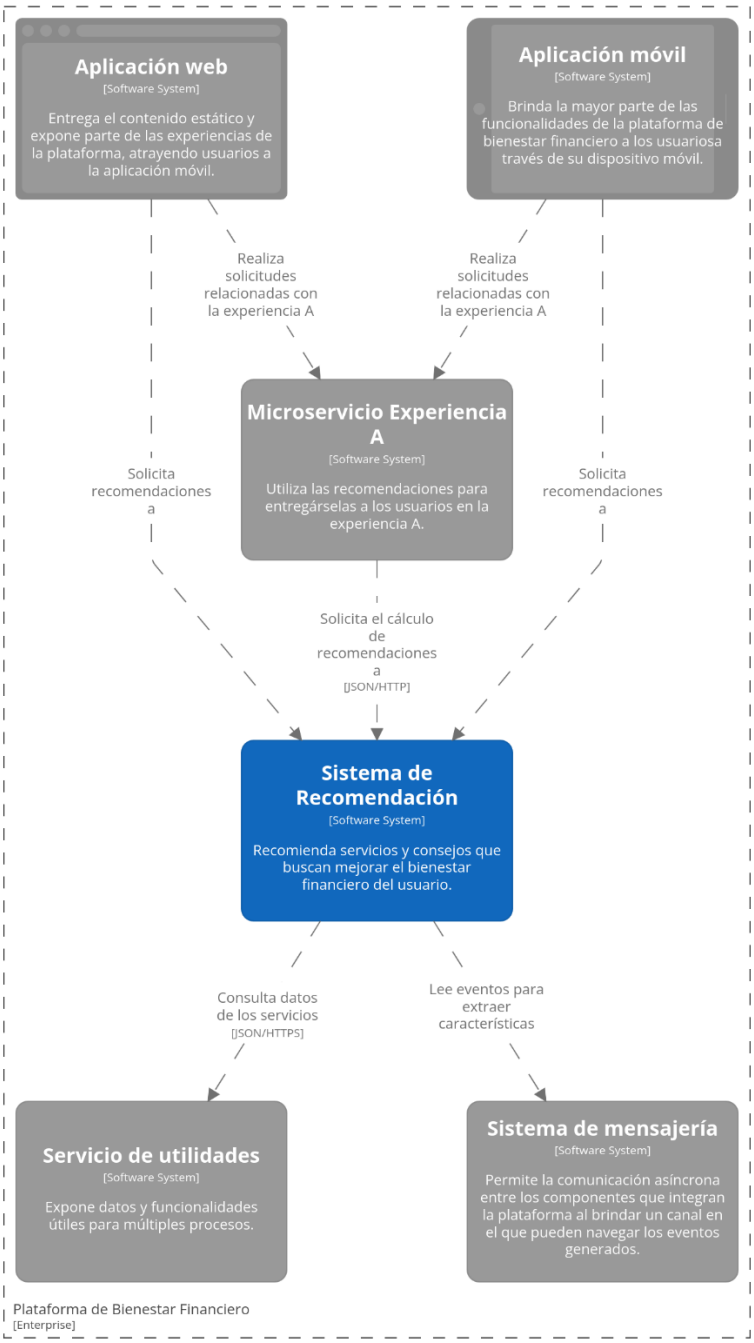


Fig. 6: Sistema de recomendación en el contexto de la plataforma.

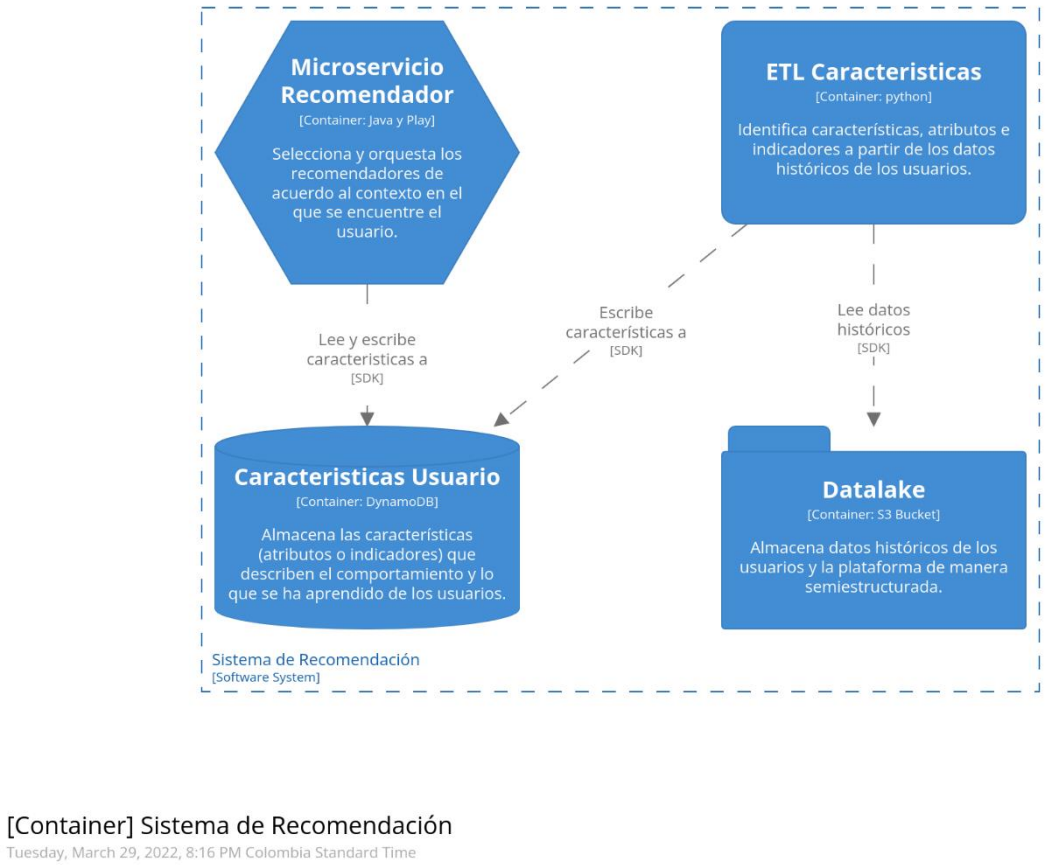


Fig. 7: Procesamiento offline de características

El Microservicio Recomendador es la implementación del orquestador mencionado en la figura 5. Este expone los servicios que permiten a los clientes poder acceder a las recomendaciones mediante solicitudes. Las solicitudes pueden contener datos, los cuales son procesados por el Procesador de datos, el cual es un componente que tiene la capacidad de interpretar los datos que vienen de cada contexto y convertirlos en características. Además de las características calculadas en tiempo real, el Microservicio Recomendador tiene a su disposición el repositorio de Características Usuario, un feature store donde se encuentran consolidadas las características calculadas en procesos tanto online como offline. Por cada solicitud, el Microservicio Recomendador identifica los recomendadores adecuados para el contexto que haya

realizado la respectiva solicitud, recupera las características del usuario, las pasa como parámetros a los modelos de recomendación, y recibe el resultado que es sometido a procesos de filtrado y enriquecimiento de contenido estático, para finalmente ser entregado al usuario. Los componentes e interacciones involucradas en este proceso son descritos en la figura 10.

En el Microservicio Recomendador se encapsulan las funcionalidades en servicios de dominio, donde se implementa la lógica de dominio, como los criterios para seleccionar un recomendador según un contexto determinado o la información que se posea de un usuario en un momento dado. En la figura 8 se observan dos servicios: el servicio de características, que permite leer y escribir características en el repositorio de Características Usuario a través de un cliente a la medida; y el servicio de recomendaciones, el cual recupera las características de un usuario usando al servicio mencionado anteriormente, e invocando al recomendador adecuado pasándole las características como parámetros. Los diferentes recomendadores son encapsulados por la interfaz Recomendador, permitiendo así que la invocación de los diferentes recomendadores sea transparente para el servicio de recomendaciones, independientemente de las técnicas con las que haya sido implementado el recomendador usado.

El proceso del cálculo de recomendaciones online es mostrado en la figura 9: un cliente, en este caso un microservicio, solicita el cálculo de recomendaciones pasando los datos de su dominio en el cuerpo de su solicitud al Microservicio Recomendador; esta solicitud es recibida por el controlador de recomendaciones, el cual le solicita al servicio de recomendaciones el cálculo de recomendaciones para el usuario; el servicio identifica el recomendador adecuado dada la solicitud de recomendaciones; luego recupera las características del usuario, tanto aquellas que ya hayan sido calculadas, como las que puedan ser calculadas de los datos proporcionados en la solicitud. Con las características del usuario y el recomendador identificado, el servicio de recomendaciones procede a invocar al modelo de recomendación, consulta los servicios que ha contratado el usuario para filtrarlos de la lista de recomendaciones, y retorna las recomendaciones al controlador de recomendaciones, que finalmente las entregara al cliente que realizo la solicitud originalmente.

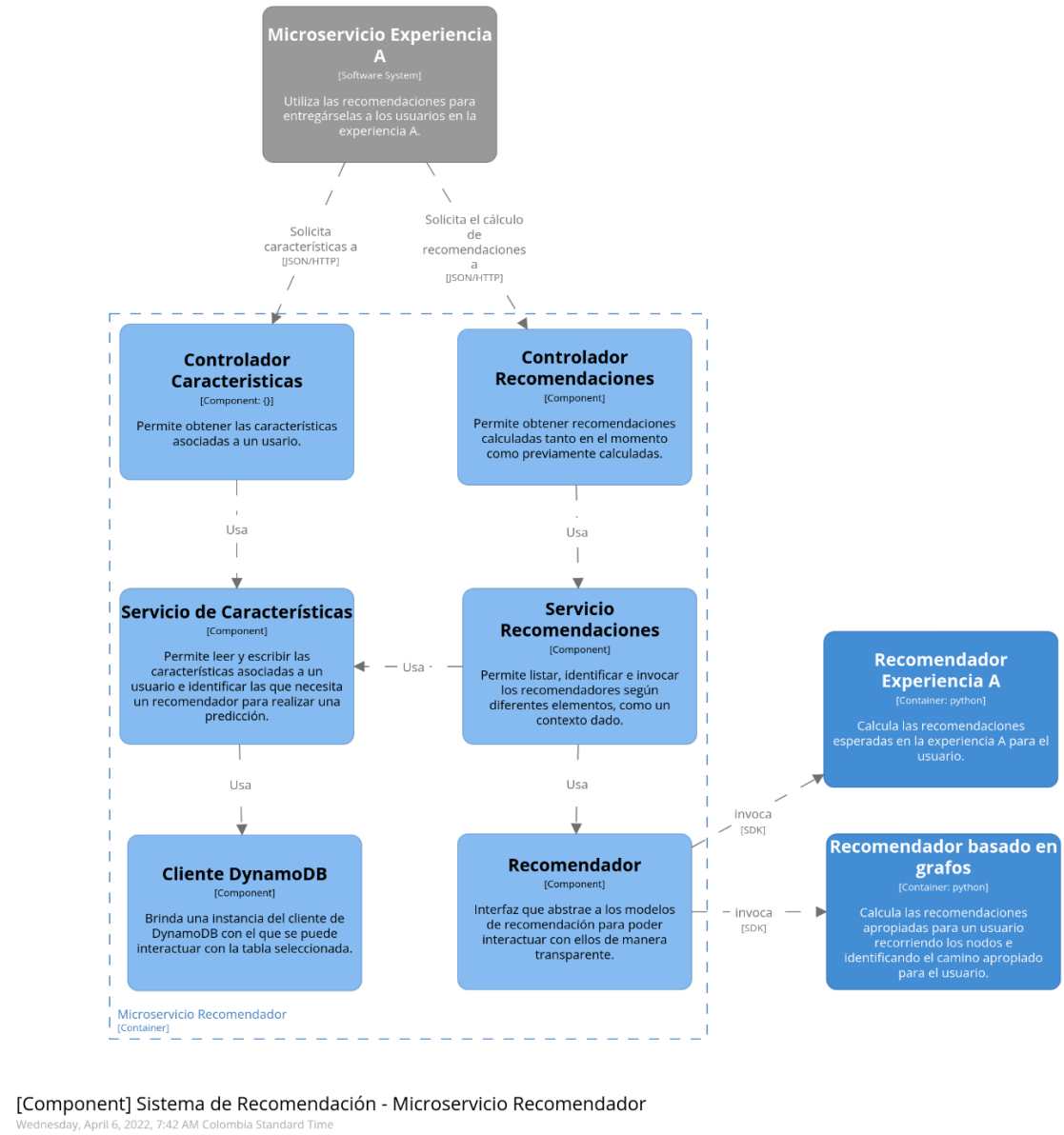


Fig. 8: Componentes en el Microservicio Recomendador

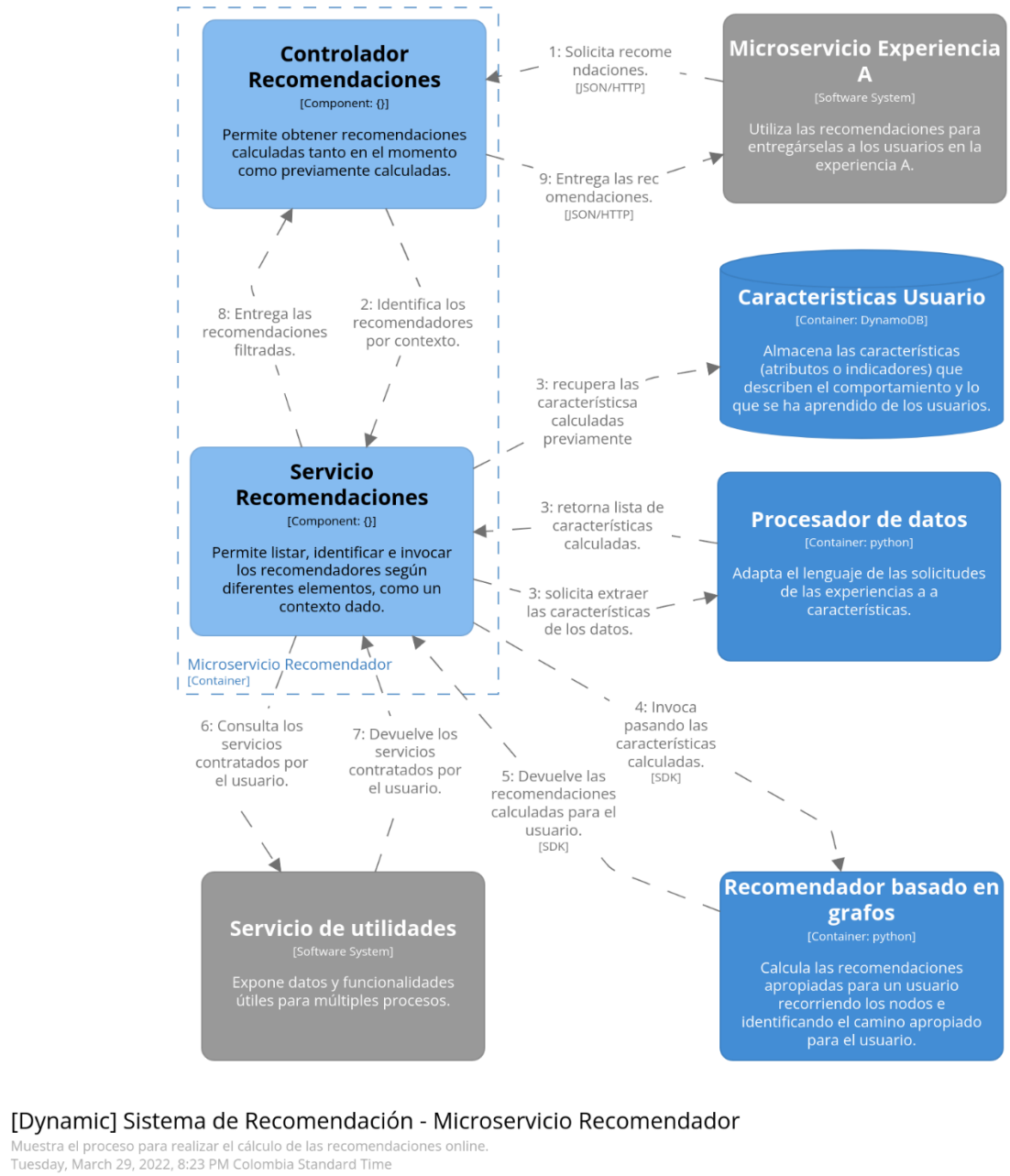


Fig. 9: Flujo en el cálculo de las recomendaciones en línea

Al migrar del recomendador legado al Proceso de Recomendación se facilitó la interacción entre los microservicios que usan las recomendaciones, debido a que la plataforma utiliza un service mesh que habilita la comunicación directa entre los microservicios en una red

privada para las instancias, mejorando la velocidad de comunicación, entre los servicios, al no tener que realizar consultas a través del servicio API Gateway, que implica que los paquetes deban salir a una red externa menos optimizada para establecer la conexión entre ellos.

De manera similar, la plataforma cuenta con una librería base, transversal a todos los microservicios, en la que se incluyen clientes que estandarizan la ejecución de solicitudes entre los microservicios. Poner a disposición en esta los Data Transfer Object (DTO) necesarios para poder realizar la solicitud de recomendaciones evita que futuras integraciones con el Microservicio Recomendador deban repetir estas implementaciones, mejorando la productividad de los desarrollos asociados a las recomendaciones.

La integración del recomendador basado en grafos y los filtros creados a partir de la información disponible, en otros microservicios de la plataforma, ayudaron a mejorar la percepción general de las recomendaciones recibidas por los integrantes del equipo de trabajo. La creación de una interfaz única para invocar los recomendadores habilita una experimentación constante entre diferentes versiones de los recomendadores, sea ajustando los existentes o creando nuevos modelos que puedan mejorar el desempeño de las recomendaciones entregadas.

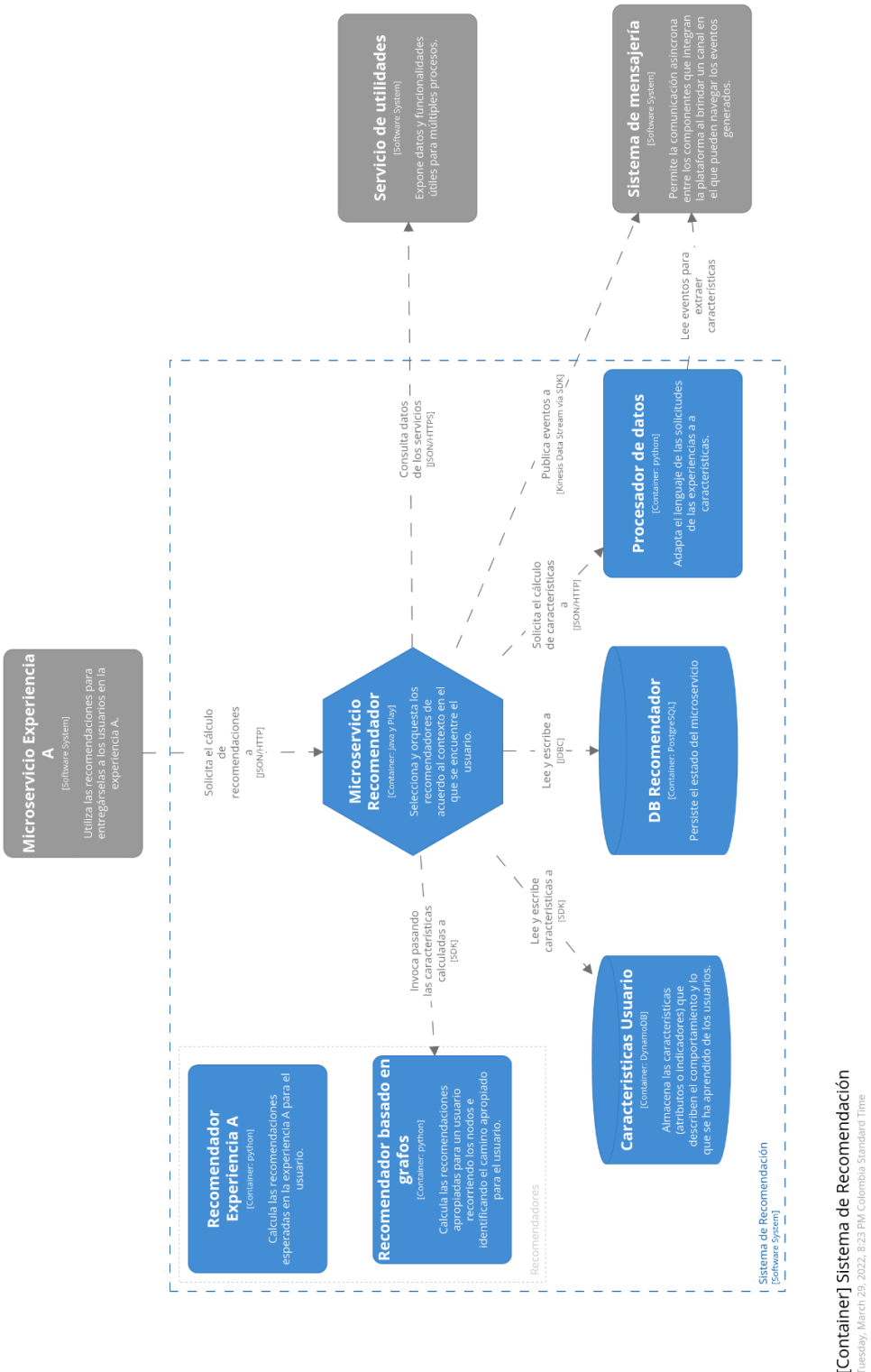


Fig. 10: Componentes que intervienen en el proceso en línea de recomendaciones

VI. CONCLUSIONES Y TRABAJO FUTURO

Las abstracciones presentadas permitieron la estandarización del cálculo de recomendaciones, consolidando la información de múltiples fuentes de datos de naturaleza síncrona y asíncrona en características, que pueden ser usadas para invocar recomendadores independientemente de las técnicas con las que estos estén implementados, de manera transparente para cualquier cliente en la plataforma. Crear diagramas a diferente nivel de detalle, haciendo uso del modelo C4 como herramienta de documentación de la arquitectura, permite seleccionar el diagrama adecuado según el público y el espacio, permitiendo ilustrar, agilizar y puntualizar las discusiones alrededor de la arquitectura diseñada.

El microservicio que orquesta el proceso de recomendación obedece a los estándares de arquitectura empresarial de la plataforma, lo que facilita la interoperabilidad con el resto de los sistemas y permite aprovechar funcionalidades transversales a toda la plataforma.

La implementación realizada está sujeta a evoluciones contempladas desde la fase, que se planean incorporar en releases posteriores, como crear mecanismos para calificar las recomendaciones y medir el desempeño de los modelos de recomendación, invocar modelos de recomendación como consecuencia de un evento o entregar recomendaciones a un usuario en canales diferentes adecuando la presentación de cada una al medio.

REFERENCIAS

[1] V. Côxo Rocha, «Infrastructure and Recommendation System for Banking Products,» Técnico Lisboa, Lisboa, 2018.

[2] Amazon Web Services, Inc., «What is AWS?,» [En línea]. Available: https://aws.amazon.com/en/what-is-aws/.

[3] J. Lewis y M. Fowler, «Microservices: a definition of this new architectural term,» [En línea]. Available: https://martinfowler.com/articles/microservices.html.

[4] M. Fowler, «BoundedContext,» [En línea]. Available: https://martinfowler.com/bliki/BoundedContext.html.

[5] M. Fowler, «CQRS,» [En línea]. Available: https://martinfowler.com/bliki/CQRS.html.

[6] C. Richardson, «Domain Event,» [En línea]. Available: https://microservices.io/patterns/data/domain-event.html.

[7] M. Fowler, «Event Sourcing,» [En línea]. Available: https://martinfowler.com/eaaDev/EventSourcing.html.

[8] X. Amatriain y J. Basilico, «System Architectures for Personalization and Recommendation,» [En línea]. Available: https://netflixtechblog.com/system-architectures-for-personalization-and-recommendation-e081aa94b5d8.

[9] Cloud Architecture Center, «Minimizing real-time prediction serving latency in machine learning,» [En línea]. Available: https://cloud.google.com/architecture/minimizing-predictive-serving-latency-in-machine-learning.

[10] Hopsworks, «What is a Feature Store?,» [En línea]. Available: https://www.featurestore.org/what-is-a-feature-store.

[11] S. Brown, «The C4 model for visualising software architecture,» [En línea]. Available: https://c4model.com/.

[12] Scrum.org, «What is Scrum?,» [En línea]. Available: https://www.scrum.org/resources/what-is-scrum.

[13] PlayFramework , «Play Framework - Build Modern & Scalable Web Apps with Java and Scala,» [En línea]. Available: https://www.playframework.com.

[14] M. Fowler y P. Sadalage, «Evolutionary Database Design,» [En línea]. Available: https://martinfowler.com/articles/evodb.html.

[15] D. Dietrich, «Vavr,» [En línea]. Available: https://www.vavr.io/.

[16] Junit.org, «JUnit 5,» [En línea]. Available: https://junit.org/junit5.

[17] Site.mockito.org, «Mockito framework site,» [En línea]. Available: https://site.mockito.org.

[18] M. Fowler, «TestDouble,» [En línea]. Available: https://martinfowler.com/bliki/TestDouble.html.

[19] Sonarcloud.io, «Automatic Code Review, Testing, Inspection & Auditing | SonarCloud,» [En línea]. Available: https://sonarcloud.io.

[20] C. Richardson, «Microservices Pattern: Database per service,» [En línea]. Available: https://microservices.io/patterns/data/database-per-service.html.

[21] N. Ford y M. Richards, Fundamentals of Software Architecture: A Comprehensive Guide to Patterns, Characteristics, and Best Practices, O'Reilly Media, 2020.