FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE COMPUTACIÓN Y SISTEMAS

DESTRUCCIÓN DE UN MONOLITO EN MICROSERVICIOS EN FANDANGO LATINOAMÉRICA

PRESENTADO POR

SEBASTIAN BURGOS DÍAZ

TRABAJO DE SUFICIENCIA PROFESIONAL

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE COMPUTACIÓN Y SISTEMAS

LIMA – PERÚ

2021

USMP | FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE COMPUTACIÓN Y SISTEMAS

DESTRUCCIÓN DE UN MONOLITO EN MICROSERVICIOS EN FANDANGO LATINOAMÉRICA

TRABAJO DE SUFICIENCIA PROFESIONAL

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE COMPUTACIÓN Y SISTEMAS

PRESENTADO POR

BURGOS DÍAZ, SEBASTIAN

LIMA – PERÚ

2021

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de suficiencia profesional consiste en el desacoplamiento de las funcionalidades de un software en múltiples softwares independientes para la empresa Fandango Latinoamérica, esto debido a la saturación por la alta cantidad de concurrencia de usuarios que tenía el software, lo cual producía que se le aumente más capacidad computacional al servidor para evitar una caída del sistema llegando a que los costos sean muy elevados. El proyecto tiene como objetivo migrar progresivamente el sistema de Fandango a microservicios para administrar y gestionar proactivamente los recursos tecnológicos y económicos que se le da a cada servicio. La metodología utilizada primero identifica las dependencias entre los servicios y se inicia la migración desde los cuales no tengan dependencias hasta llegar a los servicios que tienen alta dependencia con otros servicios (Dehgani, 2018), así mismo se usó la metodología Scrum para la gestión del proyecto, también se usó Test-driven development como estrategia de programación y auto-escalamiento predictivo como estrategia de escalamiento de la infraestructura. Los resultados obtenidos fueron lograr una alta disponibilidad de la plataforma, reducción de costos en un 36.05% y tener una arquitectura distribuida en microservicios.

Palabras clave: Monolito, Microservicios, Desacoplamiento

ABSTRACT

The present report of professional sufficiency consists of the decoupling of the features of a software in multiple software by Fandango Latin America, the problem was in the server's saturation because it was getting high amount of users' concurrency, which produced that it was increase more capacity to the server and reaching very high costs. The project's goal is to proactively manage the tech and economic resources that are given to each service. The methodology used first identifies the dependencies among the services and begins the migration from the services without dependencies until reaching the services that have high dependency to other services (Dehgani, 2018), Scrum as methodology for project management, Test-driven development as programming strategy and predictive auto-scaling as scaling strategy. The results obtained achieving a high availability of the platform, cost reduction in 36.05% and having a distributed architecture in microservices.

Keywords: Monolith, Microservices, Decoupling

INTRODUCCIÓN

El presente informe se refiere a la destrucción de un monolito en microservicios, que se puede definir como separar un software en múltiples softwares independientes y agnósticos entre sí con el fin de que cada uno pueda escalar sus recursos computacionales según su propio uso (Fowler & Lewis, 2014).

La construcción de un software es un proceso que tiene un inicio y un fin, que se compone de el modelamiento del negocio, requerimientos, análisis y diseño, implementación, pruebas y empaquetado (distribución o despliegue), es un proceso cíclico de constantes mejoras y aumento de funcionalidades (Dávila, 2019). Tradicionalmente en el desarrollo de sistemas de información se contemplan todas las funcionalidades en un solo software donde se utilizan arquitecturas y patrones de diseño para modular el proyecto, a esto se le conoce como arquitectura monolito, donde toda la aplicación está en un simple código base que incluye múltiples servicios (Al-Debagy & Martinek, 2018)

Empresas como Netflix pasaron de tener 34.24 millones de suscriptores de paga en el primer trimestre del 2013 a más de 207.64 millones en el primer trimestre del 2021 (Stoll, 2021). Bajo este contexto se han desarrollado nuevos paradigmas para la

construcción de software que puedan escalar fácilmente ante la demanda de nuevos usuarios, y también para tener procesos de desarrollo mucho más rápidos y de escala global (Macedo, 2020). En una arquitectura de microservicios se logra tener un mejor performance en términos de rendimiento cuando se tiene una gran cantidad de peticiones (Singh & Peddoju, 2017).

El software de Fandango Latinoamérica tuvo muchos cambios a lo largo del tiempo, pero todas ellas bajo la arquitectura monolito, hasta que el costo en servidores llegó a ser muy elevado porque cada vez que se necesitaba aumentar en recursos computacionales estos encarecían los gastos en infraestructura para soportar todo el código base de la plataforma. Asimismo, si es que sufríamos alguna caída en algún servicio, era muy probable que todo el software se cayera y se detengan las ventas de entradas al cine. Esto era un problema grave, porque le brindábamos el mismo servicio a los cines para que ellos puedan vender sus entradas por su Web usando nuestra tecnología y afectaba nuestra relación con ellos, sobretodo porque estas caídas eran más probables que sucedan en fechas importantes como el día del cine o alguna preventa o estreno de una película taquillera, aunque no era un problema frecuente, éramos conscientes de que poco a poco nos acercábamos a los límites computacionales que nos ofrecía Amazon Web Services y los costos seguirían subiendo considerablemente.

En este informe, desarrollo los motivos por los cuales Fandango Latinoamérica decide migrar progresivamente el monolito a una arquitectura de microservicios, como también describo la metodología y los procesos que utilizamos para realizar la migración. En este proyecto cumplo el rol de líder y administrador del proyecto, como también pude programar algunos microservicios.

El objetivo general del proyecto es migrar progresivamente el sistema de Fandango a microservicios para administrar y gestionar proactivamente los recursos tecnológicos y económicos que se le da a cada servicio. Así mismo, los objetivos específicos son:

- Reducir los costos de infraestructura en Amazon Web Services en 25%.

- Mantener el tiempo en línea del servicio de venta de entradas en al menos 99.999%.

- Migrar las funcionalidades y módulos del monolito a microservicios.

- Diseñar políticas de escalamiento según el tráfico que recibe la plataforma y anticiparnos a picos de peticiones hacia los servidores.

El trabajo de suficiencia profesional comprende cuatro capítulos. En el primero, se describe mi trayectoria profesional y laboral de manera cronológica, englobando mis cargos, periodos, funciones, principales logros y aprendizajes. En el segundo, se menciona el contexto donde obtuve mi experiencia profesional, la evolución de la compañía a lo largo de los años y se describe información general de la institución y sus matrices, como: organigramas, divisiones, visión, misión, entre otros. En el tercero, se presenta la situación problemática y la solución desarrollada, del mismo modo se profundiza en la metodología, actividades, alcance, limitaciones, etapas y roles que llevaron a tener éxito en el desarrollo del proyecto. En el cuarto, se plantea una reflexión crítica de la experiencia profesional, considerando los aprendizajes, las responsabilidades asignadas y lo logros debido al desarrollo del proyecto. Finalmente, se describen las conclusiones y recomendaciones del proyecto.

CAPÍTULO I

TRAYECTORIA PROFESIONAL

Aprendí a programar a los 11 años y desde ese entonces me dediqué a construir distintas páginas Webs y juegos online. Luego, a los 18 años empecé mi vida profesional, partiendo de Holosens, donde se construye Cinepapaya, el cual en diciembre del 2016 fue adquirido por Fandango, una empresa de NBC Universal, donde fortalezco mis habilidades de liderazgo siendo el líder de tecnología de Fandango Latinoamérica. Durante todo este tiempo mis aprendizajes y logros fueron:

IT Lead (CTO) – Fandango
Marzo 2019 – Diciembre 2020

Líder del área de tecnología y desarrollo de producto de Fandango Latinoamérica, con presencia en 14 países de la región. A cargo de los equipos de Desarrollo (Web, Android y iOS), Infraestructura (Amazon Web Services), Project Management, Compliant, IT Help Desk, UX y UI, los cuales componían un equipo de 28 personas.

Aprendizaje:

- Liderar un equipo de 28 personas, y evitar que se generen subgrupos, fomentando la comunicación horizontal entre todos.

- Sustentar y defender iniciativas del área tanto a nivel tecnológico como a nivel personal.

- Implementar políticas, procesos y pasar por auditorías de terceros, en su mayoría, empresas ubicadas en Estados Unidos e Inglaterra.

- Gestionas las necesidades de otras áreas y de negocio para que sean priorizadas en el backlog.

Principales logros:

- Diseñar e implementar un mecanismo de innovación continua para la creación de nuevos productos/funcionalidades. Lo llamamos “Hack Days”.

- Diseñar un mecanismo de promociones y aumentos salariales donde cualquiera pueda ubicarse y entender que necesita mejorar para seguir avanzando dentro de la compañía.

- Cumplimiento de políticas y procesos de NBC Universal. Logrando pasar con éxito las distintas auditorías realizadas por NBC.

- Implementar procedimientos para hacer research constantes para mejorar la toma de decisiones sobre las funcionalidades o nuevos productos que realizamos.

- Establecer políticas y procedimientos de escalamiento de la plataforma para soportar alta demanda de tráfico. En el 2019 duplicamos las ventas y cuadruplicamos el tráfico respecto al 2018, logrando un 99.999% de up-time.

- Aprobar satisfactoriamente las certificaciones y auditorías de terceros con los cuales logramos obtener la certificación PCI.

- Implementación de metodologías para la gestión de tareas, y así equilibrar los requerimientos propios del área, de negocio y de otras áreas.

- Promover la continuación del desacoplamiento del monolito en múltiples microservicios.

- Implementación del primer roadmap del área, con el cuál se pudo planificar ordenadamente proyectos que serían desarrollados a lo largo del año.

- Encargado del área de tecnología para el proceso de liquidación de la compañía en los 14 países donde se tenía presencia.

Reporte directo al gerente general y a la gerencia de Fandango US

Integration Development Team Leader – Fandango
Diciembre 2016 – Febrero 2019

Planear, diseñar y construir nuevas funcionalidades y herramientas de gran escala en Fandango.

Aprendizaje:

- Identificar el momento y las razones para comenzar a dividir un monolito en microservicios.

- Identificar y predecir patrones de comportamiento de fraudulentos.

- Contratación de personal.

- Delegar tareas.

- Mentorear y gestionar el conocimiento.

Principales logros:

- Desacoplamiento del sistema que gestiona la conexión con las APIs de los cines a un microservicio con capacidad de activar y desactivar cines según la disponibilidad de sus servicios en tiempo real usando el patrón de diseño Circuit Breaker.

- Creación de un sistema antifraude basado en árboles de decisiones y con un motor de inteligencia artificial que detecta patrones de comportamiento fraudulento.

- Integrar nuevos cines de 6 países.

- Construir un equipo de 6 personas donde se creó una metodología para transmitir el conocimiento fácilmente entre todos.

Integration Development Team Leader – Cinepapaya
Julio 2015 – Noviembre 2016

Líder del equipo de integraciones, el cual se encargaba de conectarse con terceros como cines, procesadores de pagos y facturadores electrónicos, como también del proceso de validación de entradas.

Aprendizaje:

- Liderar y contratar a los primeros miembros del equipo de integraciones.
- Negociar funcionalidades con las áreas técnicas de los cines.
- Desarrollar software de alta disponibilidad.
- Modelos, políticas y estándares de facturación a nivel regional.
- Procesamiento de dinero en distintos países de la región.

Principales Logros:

- Integrar más de 20 cadenas de cines de la región, 8 procesadores de pago y 4 facturadores electrónicos.
- Coordinación directa con los cines.
- Seleccionar y liderar un equipo de 5 personas.
- Definir estándares de comunicación a nivel de servicios con los cines.
- Desarrollar y mantener el proceso de compra y redención de entradas.
- Desarrollar y mantener la generación de facturas y boletas electrónicas para todos los países donde Cinepapaya tenga presencia.
- Desarrollar y mantener el procesamiento de dinero.

- Asegurar continuidad en el proceso de compras de entradas al 99.999%

Software Developer – Cinepapaya
Enero 2011 – Junio 2015

Primer empleado en Cinepapaya. Construyó desde cero el proceso de compra y el sistema de carteleras, proyecto con el cual fueron seleccionados por Wayra (aceleradora de telefónica), Startup Chile (aceleradora promovido por el gobierno de Chile), los cuales permitió al startup tener una proyección internacional.

Aprendizaje:

- Desarrollar un software desde cero.

- Formar y contratar al primer equipo de la empresa.

- Conceptos de Startups: Fundación, levantamiento de capital, producto mínimo viable, product market fit, lean startup, business canvas, entre otros.

Principales Logros:

- Desarrollar la página de carteleras.

- Desarrollar la página de noticias.

- Desarrollar el proceso de compra.

- Desarrollar el proceso de validación de entradas.

- Integrar la plataforma en 12 países con procesadores de pago y facturadores electrónicos distintos.

CAPÍTULO II

CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA

Holosens SAC, es la empresa que creó Cinepapaya (luego bajo la razón social Papaya Perú SAC). Holosens funcionaba como una fábrica de software, donde se realizaban proyectos internos (Startups) y proyectos externos. Entre los proyectos internos que se desarrollaron estuvieron:

- Empresabio, marketplace donde conectaba a inversores con proyectos, donde el inversor podía revisar en detalle el estado del proyecto y sus necesidades financieras y por otro lado el proyecto podía gestionar sus finanzas y promocionar sus servicios/productos en la plataforma para conseguir financiamiento.

- Empréstame/Renegocia, marketplace que permitía a una persona con deudas subir su información crediticia para que los bancos puedan competir para ofrecerle una mejor tasa.

- Citas médicas, portal donde cualquier consultorio podía crearse una cuenta y sus pacientes puedan reservar citas y pagarlas.

- Emequity, portal para que cualquier persona pueda invertir en la bolsa de valores, sin necesidad de pasar por un agente de bolsa.

- Papaya, sistema de video bajo demanda con streaming adaptativo, un Netflix para Latinoamérica.

2.1 Cinepapaya

Papaya fue el proyecto que con el tiempo fue tomando más peso en la compañía y donde todos los esfuerzos se fueron enfocando. Es así como en el 2011, el proyecto comienza a construirse y en diciembre de ese mismo año es seleccionado por Wayra, aceleradora de empresas del Grupo Telefónica, siendo parte de la primera promoción de 10 startups peruanas que participarían en ese programa.

En el 2012, y durante su aceleración en Wayra, Papaya se independiza como Papaya Perú SAC y toma el nombre de Cinepapaya, dejando el negocio del streaming y enfocándose en la venta de entradas al cine, siendo UVK – Caminos del Inca el primer cine con el que empiezan a vender entradas en abril 2012. Ese mismo año, Cinepapaya es seleccionada por el programa de aceleramiento del gobierno chileno, Startup Chile, siendo así la primer Startup peruana en llegar al país del sur. En junio de ese mismo año, Cinepapaya hace un soft-landing en Colombia y Chile.

El 2013 y 2014 fueron los años de consolidación para Cinepapaya, donde se expandió a varios países de la región vendiendo entradas para las principales cadenas de cada país, como también expandió su presencia en carteleras en los 5 continentes. El equipo pasó de ser 5 personas a poco más de 15 y pasó a tener equipos en Colombia y Chile.

En el 2015, la empresa masifica sus operaciones llegando a 12 países de la región y recibe la inversión más grande hasta ese momento en la Latinoamérica, USD$ 2 000 000 de parte de la empresa brasileña Movile. Con esta inversión, el equipo se muda a nuevas oficinas y pasa de 15 empleados a más de 40.

Ese mismo año Cinepapaya toma la iniciativa de crear nuevos verticales de negocio basadas en su tecnología de ticketing, es así como nacen:

- Papayapass, venta de entradas a teatros, conciertos y eventos en general. Se vendían todos los eventos promovidos por Movistar y se trabajo la primera alianza que tuvo la Federación Peruana de Fútbol para la venta de entradas exclusivamente online a través de la modalidad de Golden Pass.

- Papayabus, venta de pasajes terrestres con la mano de Cruz del Sur.

- Papayafast, sistema para que los restaurantes permitan a sus clientes solicitar sus servicios vía delivery, recojo en tienda o atención en mesa. Se trabajo en conjunto con Pardo’s Chicken.

En el diciembre del 2016, Cinepapaya hace un Exit y vende la compañía a Fandango Media LLC, una subsidiaria de NBC Universal. Desde ahí Cinepapaya cambia de nombre a Fandango Latinoamérica y asegura su expansión internacional. Implementaron nuevos procesos, políticas y estándares, como también el crecimiento del equipo de 40 empleados a casi 80, ocupando 3 oficinas en el edificio Link Tower ubicado en Manuel Olguin 335 – Santiago de Surco.

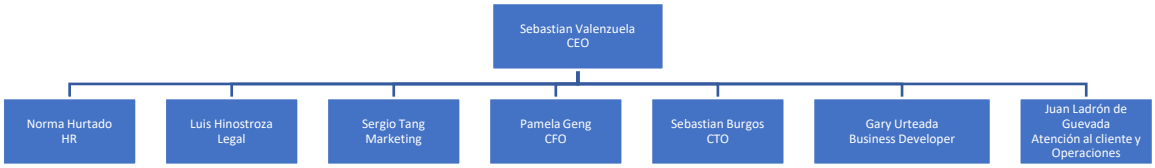

Figura 1. Organigrama Fandango

Elaboración: El autor

Visión

Ser el medio digital líder en tráfico en la industria del cine en Latinoamérica, valorada por su talento humano, innovación y tecnología.

Misión

Brindar la mejor experiencia digital de consumo de contenido y de compra de entradas al cine generando valor al público, exhibidores, estudios, anunciantes y socios comerciales de Latinoamérica.

Valores

- Excelencia
- Compañerismo
- Compromiso
- Pasión

Evolución del software

El proyecto se inició en el 2011 siendo un sistema de video bajo demanda con streaming adaptativo, el cual calculaba en tiempo real la velocidad de tu internet y dependiendo de eso distribuía en tiempo real una resolución distinta para que la reproducción del video nunca se interrumpa, tenías calidades como 144p, 240p,

360p, 480p, 720p y 1080p. Esta tecnología estaba soportada por Wowza, el cual es un motor de streaming basado en software libre.

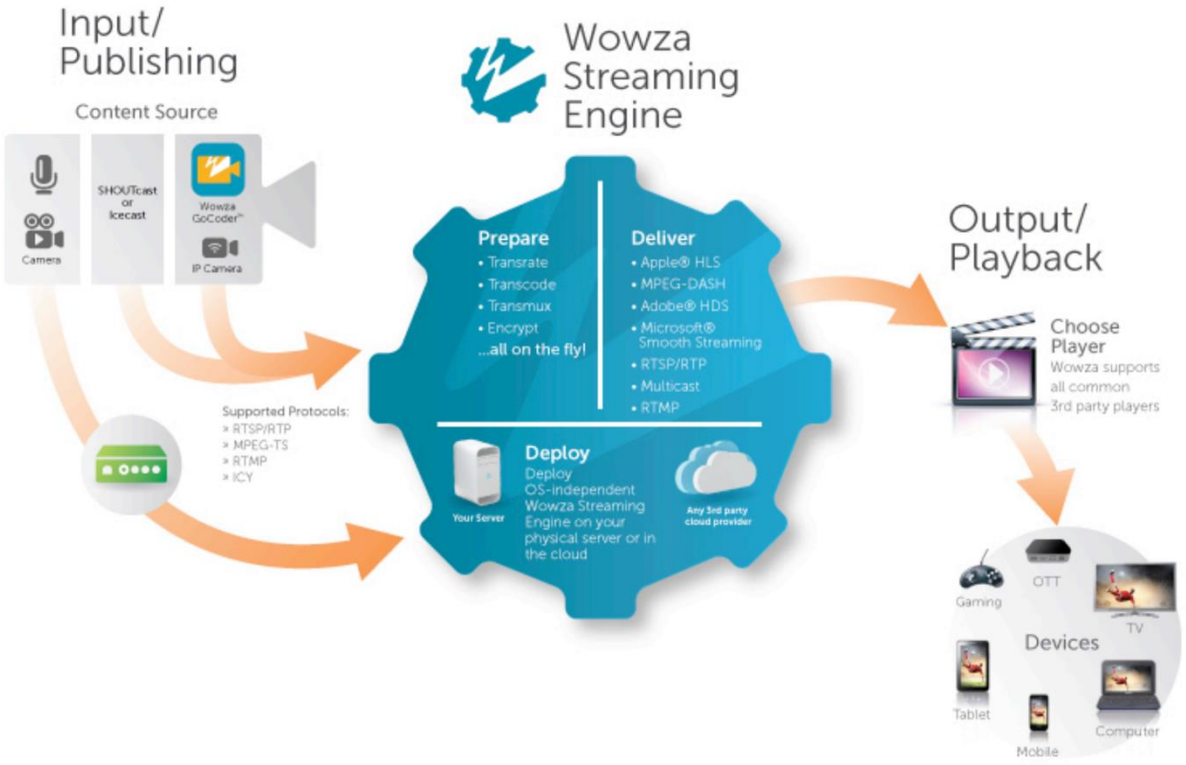

Figura 2. Arquitectura de Wowza Streaming Engine

Elaboración: *(Wowza, 2014)*

A la par se desarrolló un sistema de carteleras en papaya.pe, desde el cual buscaba posicionar la marca Papaya como un referente en cine y luego la transición al sistema de streaming sea muy natural.

El problema estuvo cuando Netflix ingresa a Latinoamérica, con lo cuál Papaya tuvo que hace un pivot a la venta de entradas al cine. Se pudo mantener la tecnología, basada en PHP, y se extendió para su soporte de venta de entradas al cine, conectándose vía Web Services con UVK y con Stripe para el procesamiento de dinero.

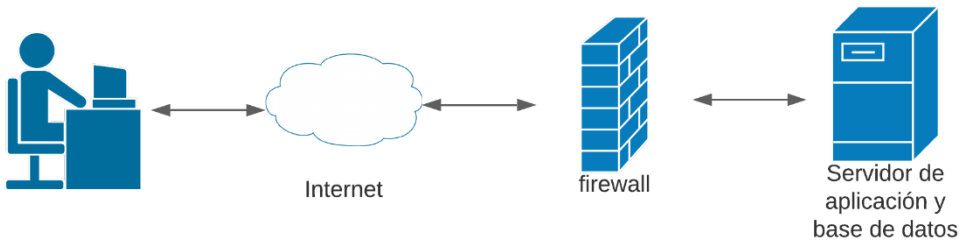

Figura 3. Arquitectura Cinepapaya

Elaboración: El autor

En el 2012, se firma con Cinemark y eso produjo el replanteo de la aplicación, y se rediseñó el proceso de compra utilizando el patrón de diseño *abstract factory*, con el cuál se pudo abstraer la lógica que interactuaba con los cines de la lógica principal del sistema, haciendo que la lógica central sea agnóstica al cine con el que se conecta, permitiendo en un futuro poder conectarse con otros cines.

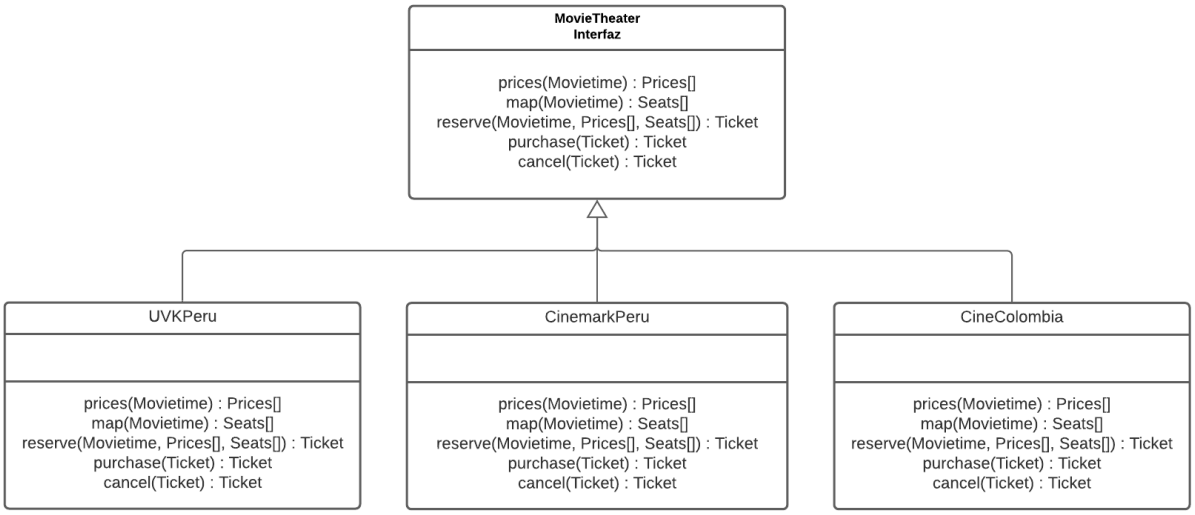

Figura 4. Abstract Pattern

Elaboración: El autor

Ese mismo año, Papaya cambia de nombre comercial a Cinepapaya y consolida su expansión internacional en Chile y Colombia gracias a Startup Chile, siendo la primer Startup peruana en ingresar al prestigioso programa de aceleración

promovido por el gobierno de Chile. Para esta expansión se tuvo que rediseñar el software para que soporte internacionalización y localización, el cuál permite el manejo de múltiples países, monedas, idiomas, modismos, huso horario y formato de fechas y horas. Del mismo modo, se integraron nuevos cines, pasarelas de pago y facturadores electrónicos.

Figura 5. De Papaya a Cinepapaya

Elaboración: El autor

La demanda en tráfico siguió creciendo, la plataforma se migró de *Codero VPS* a *Amazon Web Services*. Asimismo, se volvió de reprogramar la plataforma para soportar varias capas de datos y poder tener un Content Delivery Network delante de la plataforma para soportar altos picos de demanda en tráfico, como también para poder tener bases de datos no relacionadas como caché y así aligerar la carga de peticiones que pueda recibir la base de datos transaccional. Este fue el último gran rediseño de la plataforma, a partir de ahí se fue desagregando el software en múltiples microservicios, los cuales nacían con el concepto de ser agnósticos al resto de la plataforma, esto permitía poder desacoplarlos, reemplazarlos o apagarlos sin que afecte al resto de los servicios.

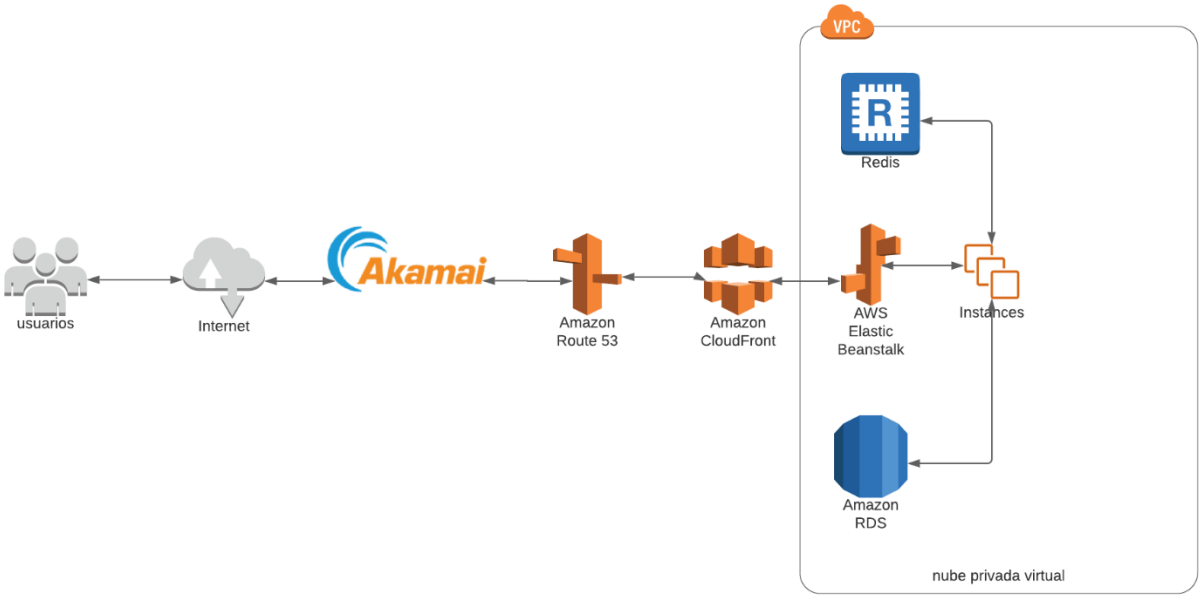

Figura 6. Arquitectura Base Monolito

Elaboración: El autor

Es, en esta última etapa del desacoplamiento del monolito a microservicios donde se centra el presente trabajo. Mi rol en esta etapa estuvo en la conceptualización de la metodología para el desacoplamiento como también en definir cuales serían estos servicios, priorizarlos y colocarlos en la hoja de ruta del área de tecnología por cada año. Desde un inicio, sabíamos que no iba a ser un trabajo de meses, porque debíamos equilibrar las necesidades del área con las necesidades del negocio.

El uso de microservicios conlleva nuevas complejidades en la construcción, despliegue y mantenimiento. Por lo cual, definimos un orden de prioridades basado en funcionalidades que en ese momento nos estaban generando problemas en la estabilidad de la plataforma o en funcionalidades que consideremos que debíamos reforzar la seguridad o en funcionalidades que en un futuro podían ser problemáticas. Basado en eso, definimos la prioridad de cada una, considerando la dependencia que podrían tener entre ellas y en evitar redundancia en los servicios. Se creó un mapa con estos servicios, se definió las tecnologías más adecuadas

para cada solución y se negoció con las áreas de negocio para incluir estos proyectos a lo largo de los años.

En el Capítulo III, se dan más detalles de todo este proceso.

Cines:

- Chile
  - Cine Hoyts
  - Cinestar

- Argentina
  - Showcase
  - Village Cines
  - Atlas Cines
  - Cinema La Plata
  - Sunstar Cinemas
  - Cinema Adrogue
  - Cinema Casey
  - Cinema Pergamino

- Bolivia
  - Cine Center
  - Cinemark
  - Multicine

- Perú
  - Cinemark
  - UVK
  - Cinestar
  - Movietime
  - Cinépolis

- Ecuador
  - Cinemark
  - Cinext
  - Multicines
  - Cineplex
  - Mis Cines

- Colombia
  - Cine Colombia
  - Royal Films
  - Procinal Bogota
  - Procinal Medellin
  - Cineland
  - Cinema Paraiso
  - Cinépolis

- Panamá
  - Cinemark
  - Cinemas Royal

- Costa Rica
  - Cinemark

- Guatemala
  - Cinemark

- El Salvador
  - Cinemark

- Honduras

    - Cinemark
    - Metro Cinemas

- República Dominicana
    - Palacio del Cine

- Nicaragua
    - Cinemark
    - Cinemas

- México
    - Cinebox
    - Cinemagic
    - Cinemex
    - Cinetix

2.2 NBC Universal Media, LLC

Es una empresa estadounidense enfocada en medios de comunicación masivos, fue formada en el 2004 luego de la fusión de National Broadcasting Company (NBC) y Vivendi Universal Entertainment. La cuál fue en el 2009 adquirida por Comcast, el conglomerado de medios de comunicación.

Misión

Ser el principal proveedor de contenido para televisión y plataformas digitales, abarcando toda la televisión.

Visión

Ascender como un actor principal en la industria de los medios y el entretenimiento.

Las empresas que son parte de NBC Universal son:

1. División de transmisiones:
    - NBC
    - NBC News
    - NBC Sports
    - Telemundo
    - NBC Olimpiadas

2. Redes de cable:
    - Golf
    - Canal de las Olimpiadas
    - Oxygen
    - Universal Kids
    - USA
    - SY FY

- MSNBC
- CNBC
- Bravo
- E! Entertaiment
- NBCSN

3. Negocios Digitales
    - GolfNow
    - Sport Engine
    - **Fandango**

4. Grabaciones
    - Universal Pictures Home Entertainment
    - DreamWorks
    - Focus Feature
    - Universal

5. Internacional
    - Hayu
    - CNBC International
    - Universal Pictures International

6. Medios Locales
    - NBC Sports Regional Networks
    - NBCUniversal Owned Television Stations
    - Telexitos
    - Cozi TV

7. Parques y Resorts
    - Universal Studios Hollywood
    - Universal Orlando Resort

- Universal Studio Japan
- Universal Studios Singapur
- Universal Studios Beijing Resort

8. Streaming
   - Peacock

9. Producción de TV
   - Telemundo Studios
   - UCP
   - Universal Television

Organigrama

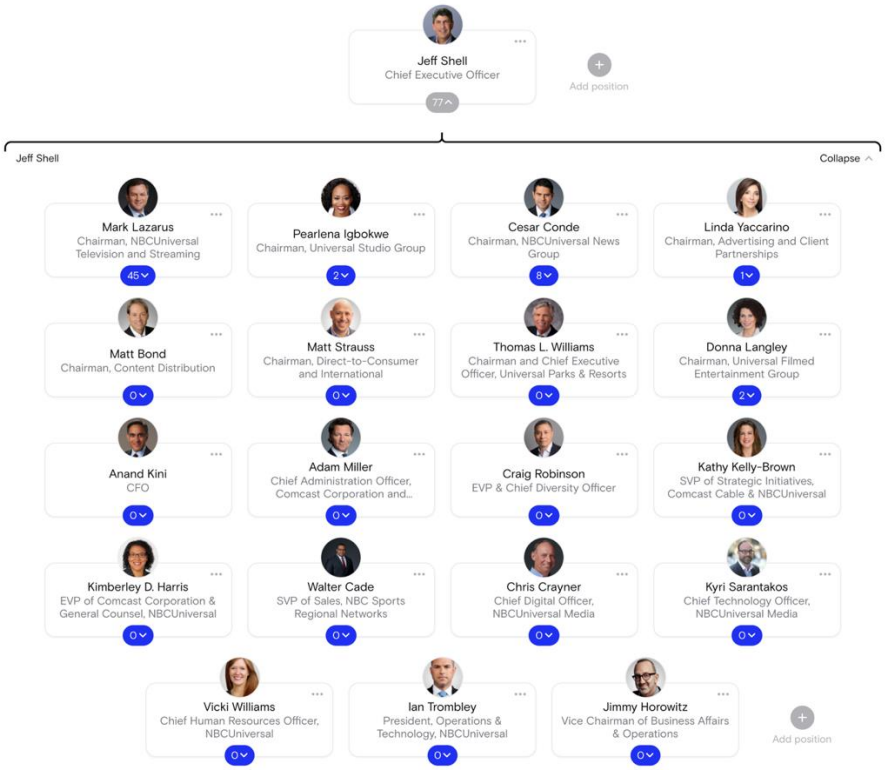

Figura 7. Organigrama de NBC Universal

Elaboración: *(The Org, 2021)*

2.3 Fandango Media LLC

Es una empresa estadounidense fundada en el año 2000 por un conglomerado de cines para la venta de entradas al cine vía telefónica, años después se convierte en un sistema de ventas de entradas al cine online. Luego, en el 2007, fue adquirida por Comcast/NBC Universal y en el 2016 el 30% de la compañía fue adquirida por Time Warner's Warner Bros. Entretainment, ese mismo año Fandango adquiere a Cinepapaya, y así expande sus operaciones en Latinoamérica, llegando en el 2020 a tener presencia en 14 países de la región.

Misión

Deleitar a los fanáticos del cine con contenido atractivo, información indispensable y herramientas innovadoras para mejorar entregue su experiencia cinematográfica total, en cualquier momento y en cualquier lugar.

Las empresas que son parte de Fandango son:

1. Fandango.com
2. Ingresso
3. Fandango Now
4. Rotten Tomatoes
5. Fandango Latam
6. Flixter
7. Vudu
8. Movie Tickets
9. Movies.com

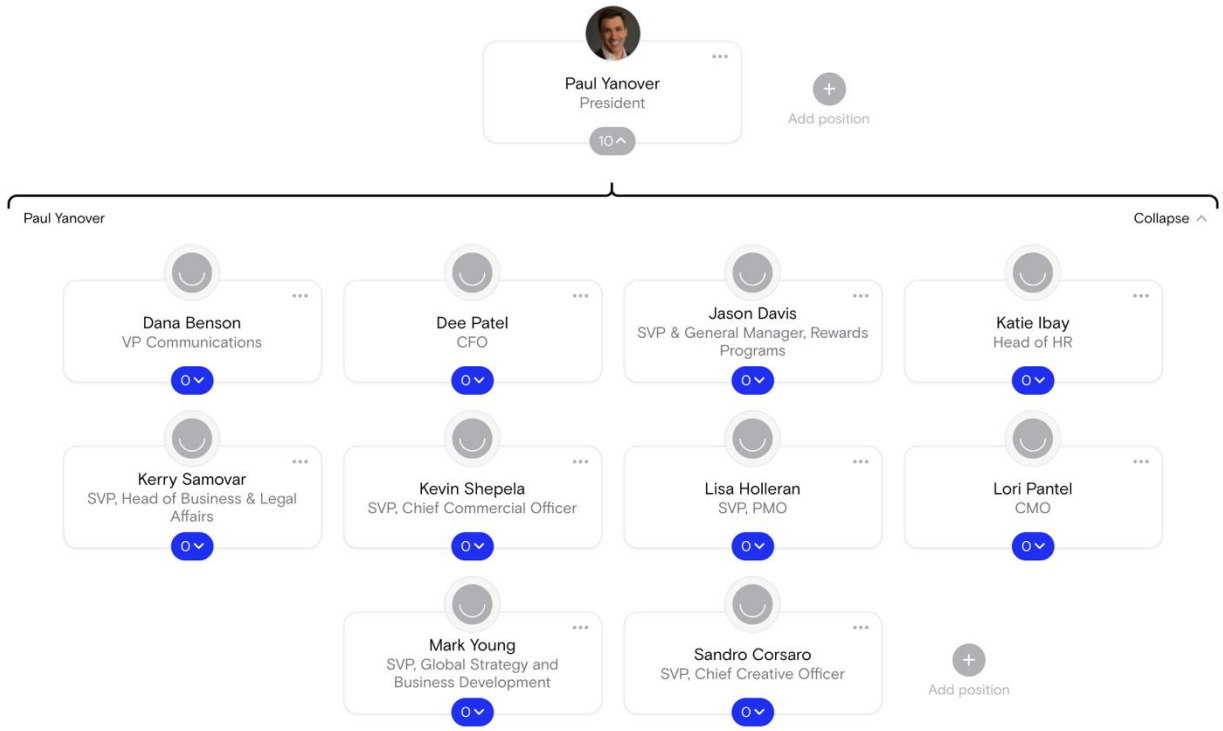

Figura 8. Organigrama de Fandango US

Elaboración: *(The Org, 2021)*

¿Por qué cierra Fandango?

Desde inicios de marzo del 2020, varios cines de la región decidieron solidariamente implementar medidas ante el reciente virus Sars-Cov-2 (Covid-19), pero en medida que cada país fue entrando a una cuarentena estricta los cines se vieron obligados a cerrar sus operaciones.

En un inicio, se veía con mucho optimismo el regreso de los cines en los próximos meses, pero en medida que las cuarentenas a nivel internacional se iban alargando algunos cines se vieron en la necesidad de endeudarse para poder sobrevivir toda esta etapa sin operaciones.

Por otro lado, los estudios de cine y productoras de películas aceleraron su crecimiento en sus plataformas de streaming como:

1. Amazon Prime
2. Disney Plus (incluido Fox Premium)
3. HBO
4. Netflix
5. Peacock de NBC Universal
6. Paramount Plus
7. Warner

Algunas productoras anunciaron que durante el 2020 y 2021 no estrenarían películas en los cines o que las estrenarían en paralelo en sus sistemas de streaming. Esto ha sido un duro golpe para la industria del cine, dada la coyuntura actual de la pandemia por el Covid-19 se ven forzados a tener un aforo limitado, adicional a eso no tener una cartelera de estrenos películas exclusivos, reduce considerablemente la cantidad de personas que estén dispuestos a ir. Es por eso por lo que en el 2020 Cinemex tuvo que cerrar 145 locales de cines indefinidamente, o que Cine Colombia recién esté planeando reabrir sus puertas en mayo del 2021. Actualmente esta crisis viene disminuyendo con algunos pocos estrenos exclusivos que, si llegaran al cine en el segundo semestre del 2021, pero con mucho miedo de parte de los cines de tener que cerrar nuevamente sus operaciones.

Dada estas dificultades en la industria, es que NBC Universal decide hacer una reestructuración de sus operaciones globales, y es ahí donde decide no continuar con las operaciones de Fandango Latinoamérica, cerrando todos sus contratos con todos los cines de la región. Solamente se quedan con las operaciones de Ingresso Brasil. Esto se anuncia en agosto del 2020.

2.4 Instacash

Desde agosto del 2020 cofundo el proyecto Instacash junto con Juan José Roca Rey Valdés y Luis Alberto Chau Tay.

Instacash es un mercado de préstamos personales garantizados con las preautorizaciones de las tarjetas de crédito. Conectamos a prestamistas y prestatarios de forma completamente anonimizada y permitimos que se presten entre ellos, el prestatario debe poner como garantía el cupo de su línea de crédito, la cuál es congelada durante el periodo que dure el préstamo y en medida que va pagando el préstamo se va liberando cada cuota de esa retención temporal.

Instacash ha sido seleccionada por diversos programas de aceleramiento, como:

1. UTEC Ventures
2. Founders Institute
3. Tetuan Valley
4. Startup Chile
5. 500 Startups

CAPÍTULO III

APLICACIÓN PROFESIONAL

Un estudio (Arboleda Cola, 2017) analiza las arquitecturas monolíticas donde menciona como inconvenientes que el escalamiento de recursos en este tipo de arquitecturas afecta a toda la plataforma generando el consumo de recursos innecesarios, cualquier problema en la aplicación pueden volverse generalizado, el software se vuelve difícil de actualizar y, por último, se vuelve difícil que distintos equipos puedan reutilizar código eficientemente.

Martin Fowler, impulsor de la arquitectura de microservicios, menciona lo siguiente: “Casi todas las historias exitosas de microservicios comenzaron con un monolito que se hizo muy grande y se rompió, y casi todas las historias que he escuchado de sistemas que se construyeron como un microservicio desde cero, han terminado en serios problemas” (Monolith First, 2015). Los microservicios siguen uno de los principios de la programación orientada a objetos, SOLID, específicamente el principio de responsabilidad única, el cual menciona: “Reúna las cosas que cambian por la misma razón, y separe aquellas cosas que cambian por diferentes razones” (Martin, 2008).

Sobre la misma línea, Newman comenta: “los microservicios son servicios pequeños y autónomos que trabajan juntos” (Newman, 2015).

Empezar con una arquitectura monolítica no está mal porque es más eficiente mientras se tenga una pequeña carga de peticiones y es una arquitectura más sencilla de desarrollar, mientras que una arquitectura de microservicios trae dificultades en la integración, conexión y configuración, pero que te asegurará soportar un mayor número de peticiones al servidor (God & Zabierowski, 2020), logrando alcanzar una reducción del costo de infraestructura hasta en un 77.08%. (Ochoa, y otros, 2016).

3.1 Antecedentes

En la construcción de las primeras versiones de Cinepapaya, se desarrolló un software monolito, el cuál comprendía todas las funcionalidades y el backend, usando PHP como lenguaje principal de la plataforma, y frontend en una sola aplicación, lo cuál para un equipo de menos de 5 personas era lo más sencillo de manejar. Se encontraba alojada en un servidor privado en EE. UU., donde aparte de tener la aplicación instalada también contaba con una base de datos MySQL.

En medida que las nuevas necesidades del negocio, como la internacionalización y localización, fueron apareciendo, se hizo necesario replantear por completo el software que se había desarrollado. Asimismo, el crecimiento de las aplicaciones móviles hizo que se comience a exponer servicios Web para que puedan consumirse, en el 2011 solo el 7.6% de la población de Latinoamérica usaba un Smartphone y para el 2014 había pasado a ser el 24% de latinos conectados desde un teléfono inteligente (eMarketer, 2017). La creación de servicios para los aplicativos móviles eventualmente fue generando que se dupliquen las funcionalidades en el software, por lo cuál generaba que se tenga funciones similares pero que devuelven datos diferentes, ya que se iba desarrollado pensando en la solución final y no en un backend que sea agnóstico al cliente que lo consuma. Como también, cada país contaba con una versión del código distinto, y como este no se encontraba versionado, cuando las diferencias entre cada

sistema fueron acentuándose, se volvía muy complejo actualizar el software en todas las plataformas a la vez, esto nos hacía triplicar el trabajo para realizar cualquier cambio que debía aplicar a más de un país.

Dado estos problemas, se reconstruyó la plataforma dividiendo el backend del frontend, construyendo servicios REST que sean agnósticos de quien lo necesite, así sea el frontend, un aplicativo móvil, un aplicativo en TV, un quiosco, entre otros dispositivos.

Del mismo modo, se separó el backoffice (panel administrativo) en otra aplicación, con su propio gestor de usuarios el cual tenía conexión a la base de datos principal para el manejo de los CRUDs que utilizaría la herramienta REST.

En el 2014, iniciamos la construcción de los servicios REST, iniciando desde cero toda la construcción del sistema considerando también internacionalización y localización, como también buscamos normalizar la lógica como el cliente se comunica con el servidor, para esto utilizamos la arquitectura *Representational State Transfer*, basado en verbos HTTP, como:

Tabla 1. Verbos HTTP REST

HTTP	CRUD	Obtener todo	Obtener elemento
GET	Obtener	201 (Creado), la cabecera 'Location' con el enlace para obtener el elemento creado.	404 (no encontrado), 409 (conflictos) si el recurso ya existe.
POST	Crear	200 (OK) Listado. Usar paginación, ordenamiento y filtros para navegar en listas grandes.	200 (OK), responde con un elemento, 404 (no encontrado) si el elemento no existe.

PUT	Actualizar/Reemplazar	405 (No permitido) a menos que quisieras que pueda editar todos los elementos de la lista.	200 (OK), 204 (Sin contenido) o 404 (no encontrado)
PATCH	Actualizar/Modificar	405 (No permitido) a menos que quisieras que pueda editar todos los elementos de la lista.	200 (OK), 204 (Sin contenido) o 404 (no encontrado)
DELETE	Eliminar	405 (No permitido) a menos que quisieras permitir que se pueda eliminar todos los elementos de la lista.	200 (OK), 204 (Sin contenido) o 404 (no encontrado)

Fuente: (Rest Api Tutorial, 2021)

REST es una arquitectura cliente-servidor en la que los servicios son recursos y se identifican mediante URLs basado en los verbos mencionados en la tabla anterior. Es una alternativa más sencilla a SOAP y fue la base con la que se construyó la Web 2.0 (Ynga & Palacios, 2015).

Utilizando este tipo de arquitectura permitió establecer un orden en la forma en que el cliente interactuaba con cada tabla de la base de datos, sea para crear, leer, actualizar o eliminar se vuelve muy predecible y normalizado cuál es el verbo HTTP que debes utilizar para cada acción.

Entre las principales características de esta arquitectura están (Fielding, 2000):

1. Arquitectura cliente-servidor
    - Permite tener una arquitectura complemente separada entre el cliente y el servidor, permitiendo que las interfaces gráficas sean agnósticas a la gestión de la lógica y el acceso de almacenamiento de información y

viceversa, asimismo permite que el servidor pueda servir a múltiples dispositivos en diferentes plataformas. Esto también implica tener escalabilidad del frontend y del backend independientemente.

2. Statelessness
    - Cada petición es tratada de manera independiente y no tiene ninguna relación con la solicitud anterior.

3. Caché
    - En cada respuesta se puede informar al cliente si necesita hacer caché de la información o no. Con esto se permite que el cliente no necesite estar consultando constantemente la misma información y así se reduce las peticiones que recibe el servidor y el cliente es mucho más rápido en responder al usuario.

4. Capas de sistema
    - Se pueden agregar capas de sistema delante del servicio Rest, por ejemplo:

        - Capa de seguridad, se puede gestionar quienes tienen acceso al servicio rest y si los tiene a que servicios puede acceder y que acciones realizar en cada uno.
        - Capa de caché, se puede tener una capa que maneje la lógica del caché de la aplicación, así reducimos la cantidad de peticiones que puede recibir la base de datos.
        - Load Balancer, en aplicaciones modernas es común tener múltiples servidores (instancias) corriendo a la misma vez, estas suelen tener por delante un balanceador de carga que define en base a un algoritmo a que instancia se va la petición que se está recibiendo, tratando siempre de mantener la misma cantidad de peticiones entre todas las instancias.

5. Código a demanda
    - Al tener las capas separadas, se puede optar por las mejores tecnologías y prácticas para cada problema.

6. Interfaz uniforme
    - Al tener una interfaz común, es sencillo desacoplar la arquitectura en múltiples capas o en otros servicios, ya que lo único que importa es la data que recibe y la que devuelve, y no lo que sucede en el procesamiento.

Esta primera versión de REST se lanzó en junio del 2015. Este servicio seguía siendo un monolito que comprendía todos los módulos del sistema como:

1. Ticketing
    - Módulo para la venta de entradas al cine, el cuál se conecta a cada uno de los cines para:

        - Obtener precios, promociones, descuentos, precios corporativos, entre otros.
        - Obtener distribución y ocupación de asientos.
        - Reservar asientos.
        - Confirmar la compra de los asientos.
        - Anular la compra.
        - Cancelar la reserva de asientos.

2. Procesamiento de dinero
    - Módulo que se conecta con los procesadores de pago de cada país para procesar el dinero localmente (lo cuál aumenta considerablemente la tasa

de aprobación). Entre las distintas formas de pago que podía procesar están:

- Transferencias bancarias: PagoEfectivo
- Billeteras Virtuales: Tigo Money
- Tarjetas de crédito/débito: NPS, Alignet, Mercadopago, Transbank, Braintree, Stripe, entre otros.

3. Generación de facturación electrónica
    - Módulo que procesa la emisión de facturas o boletas electrónicas en cada país en el que estábamos. La complejidad se encuentra en la forma que se calculan los impuestos en cada país.

4. Usuarios
    - Módulo que gestiona el registro y autenticación de usuario en la plataforma. Asimismo, permite la edición del perfil, cambio de contraseña y visualización de las transacciones realizadas.

5. Antifraude
    - Módulo que envía eventos para ciertas acciones de los usuarios en la plataforma, como también información sobre la transacción que se está procesando. Con estos datos, la herramienta antifraude podía encontrar un patrón de comportamiento y clasificar la transacción como fraudulenta.

6. Contenido de funciones
    - Módulo que permite el CRUD de funciones dentro de la plataforma, estos pueden ser de las siguientes formas:

        - Automatizado
            - Scraping – Script que ingresa a la página del cine, lee y almacena el contenido para estandarizarlo y este se

almacene en la base de datos, previa sincronización de algunos datos para homogenizar la información obtenida.

- Web Services – Con los cines que teníamos conexión para la venta de entradas, se puede consumir la información directamente desde su sistema, igual se necesita una sincronización manual como traductor entre su sistema y el de Fandango.

- Manual
  - Los cines con los que no se podía automatizar la obtención de las funciones, había que realizar un trabajo manual para ingresarlas al sistema.

7. Contenido de películas
   - Módulo para gestionar la información de una película, actores, directores, fechas de estreno, entre otros.

8. Noticias
   - Módulo para gestionar las noticias, comentarios, categorías, etiquetas, etcétera.

9. Festivales
   - Módulo para gestionar los festivales, el cuál tiene conexión con el módulo de películas y noticias.

10. Cartelera
    - Módulo que gestiona el listado de películas que van en la cartelera, el cuál también permite priorizar las películas por país.

11. Conexión con los cines

- Este módulo normaliza las comunicaciones a nivel de Web Services que se tiene con cada uno de los cines. Utiliza el patrón de diseño Abstract Factory con el cuál logramos homogenizar la entrada y salida de datos de cada uno de los cines y así cada integración sea un “plug-&-play” en la plataforma.

12. Validador de entradas
    - Módulo que recibe y envía información sobre las entradas vendidas para un local de cine en específico. De esta manera, cada vez que un lector valida una entrada puede verificarla localmente y luego enviarla al servidor central a través de este módulo para que quede marcada como validada y los otros lectores tengan la información actualizada en tiempo real.

13. Logs de aplicación
    - Módulo que abstrae los logs de aplicación y que se conecta con distintos sistemas de monitoreo para descargar la información recopilada para su analítica.

3.2 Situación problemática

A lo largo de los años, el monolito fue creciendo y se fueron parchando funcionalidades para extender el software dependiendo de los nuevos requerimientos que iban apareciendo y que no fueron contempladas en la concepción del software. Asimismo, desde el 2015 al 2019 el tráfico y ventas se multiplicaron por 10 y eso comenzó a producir problemas en latencia y eventualmente caídas del servidor donde estaba la plataforma. Estas latencias se originaban principalmente desde la conexión con los cines, dado que sus sistemas no podían escalar y en muchos casos no estaban preparados para alta concurrencia, algunos cines utilizan como servidor una computadora de escritorio común y corriente con Internet de hogar.

La arquitectura del monolito estaba centralizada en un único AWS Elastic Beanstalk, que tenía un AWS Elastic Load Balancer donde corrían 12 AWS EC2

simultáneamente, esto ocasionaba que la cantidad de concurrencia que recibía la base de datos AWS RDS MySQL era muy grande, que nos llevó a tener una base de datos maestra y otra esclava, ambos con las mismas características. La maestra sería para registrar información, pero entre la maestra y esclava se repartían la carga de consultas. Por otro lado, el procesamiento asíncrono de algunas tareas era delegado al servicio de colas AWS SQS, el cuál invocaba un AWS Lambda que luego hacía una invocación a algún servicio ubicado en el AWS Elastic Beanstalk para hacer el proceso asíncrono.

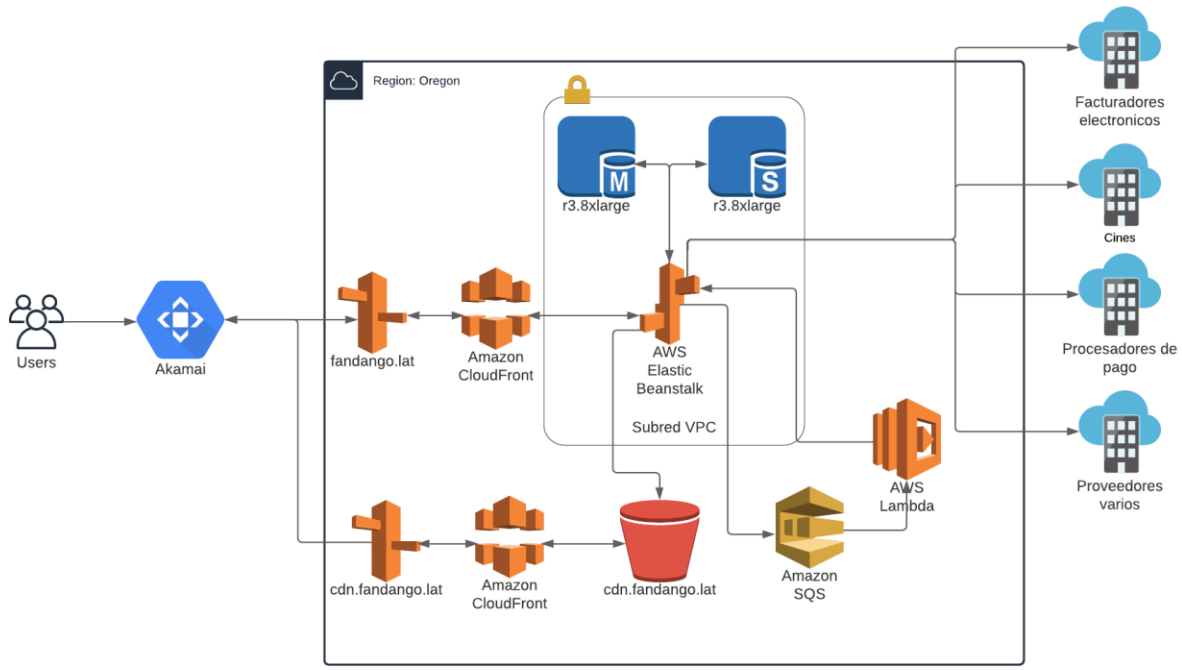

Figura 9. Arquitectura fandango

Elaboración: El autor

Durante la preventa de Avengers: Endgame, Fandango logró tener más de 3 transacciones por segundo, aunque se pudo soportar la alta demanda en ese momento, en los días siguientes algunos cines comenzaron a tener tiempos de respuesta bastante elevados y el software de Fandango mantenía la conexión abierta hasta tener una respuesta, que en cierto momento se tuvo 4 cines que presentaron este problema y la base de datos se quedó sin capacidad de recibir más peticiones, esto hizo que la plataforma se caiga por 30 minutos.

Previamente, habíamos experimentado picos en el tráfico debido a intentos de ataques de denegación de servicios distribuida en el módulo de autenticación y registros de usuarios. Lo cual en algunas oportunidades hizo que la plataforma deje de estar disponible. También identificamos ataques de fuerza bruta que ponían a nuestros usuarios en riesgo de que cierta información sea expuesta, aunque de igual forma no iban a poder realizar transacciones con solo autenticarse con algún usuario, nunca hubo un reporte de un usuario que haya identificado compras que no reconociese.

Las nuevas funcionalidades hacían que el monolito se vuelva cada vez más difícil de manejar, comprando mucha deuda técnica, la cual comenzaba a afectar en el rendimiento del equipo de desarrollo para agregar o extender funcionalidades o que sea mucho más lento para un nuevo miembro del equipo entender la complejidad que había en cada funcionalidad. Este problema es muy difícil de transmitir al negocio, poner una pausa en las necesidades del negocio para pagar la deuda técnica es un debate sin fin, pero que eventualmente se logró ir priorizando poco a poco.

Por último, mientras más íbamos creciendo, el foco en que la venta de entradas al cine esté siempre disponible se fue acentuando. Sobretodo porque muchos cines redireccionaban sus ventas hacia nosotros, en algunos cines la venta por Fandango representaba más del 50% del total de sus ventas, y en otros casos cuando se caía su sistema promocionaban el nuestro para seguir vendiendo, como Cinemark Perú en la preventa de Avengers: Endgame. Pero realmente no necesitábamos de todos los servicios para realizar transacciones, por ejemplo, podíamos no tener el servicio de noticias funcionando o inclusive el servicio de promociones, lo importante era permitir que los usuarios, sin importar de donde lleguen, puedan comprar entradas. Pero al ser un monolito era imposible enfocar los recursos en alguna funcionalidad en especial y teníamos que aumentar la capacidad en todas las funcionalidades a la vez, y eso era muy costoso para la compañía, como también que ese crecimiento tenía un tope, y estábamos muy cerca a pasarlo.

Los costos en infraestructura podían sobrepasar los 300 000 dólares al año, esto porque en momento de mucho tráfico o alta latencia de los cines, era necesario aumentar la cantidad de instancias y en su defecto aumentar la capacidad de la base de datos. Adicionalmente, no existía un plan de escalamiento y era simple percepción de cuanto se creía que debíamos aumentar.

Tabla 1. Costo mensual promedio de AWS

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
AWS Lambda	Cantidad de solicitudes (20000000), Simultaneidad (10), Hora en la que está habilitada la simultaneidad aprovisionada (720 horas)	USD$ 648.18
Amazon Route 53	Zonas alojadas (1)	USD$ 8.50
Elastic Load Balancing	Número de balanceadores de carga de aplicaciones (2)	USD$ 2 952.85
Amazon Simple Queue Service (SQS)	Solicitudes de cola estándar (5 millones por mes)	USD$ 2.00
Amazon EC2	Sistema operativo (Linux), Cantidad (12), Pricing strategy (EC2 Instance Savings Plans 1 Year No UpFront), Cantidad de almacenamiento (30 GB), Tipo de instancia (t3.2xlarge)	USD$ 1 863.34
Amazon RDS for MySQL	Almacenamiento para cada instancia RDS (SSD de uso general (gp2)), Cantidad de almacenamiento (1 TB), Cantidad (2), Tipo de instancia (db.r3.8xlarge), Opción de implementación (Multi-AZ), Modelo de	USD$ 11 605.92

	precios (OnDemand), Almacenamiento de copias de seguridad adicional (1 TB)	
	Total	**USD$ 17 080.79**

Elaboración: El autor

3.3 Definición del problema

Como se menciona en los antecedentes, el crecimiento exponencial de tráfico a la plataforma comenzó a generar que los costos en infraestructura sean muy elevados y llevando a estar cerca de los límites computacionales que ofrece Amazon Web Services, que nos llevaba a adelantarnos a un problema importante donde nos íbamos a ver sobrepasados por la demanda y sin la posibilidad de seguir escalando la plataforma. Adicionalmente, como se ha mencionado antes, el problema con una arquitectura monolítica es que cuando escalas esto aplica para toda la plataforma, y seguramente estás pagando por una gran capacidad computacional en partes del sistema que realmente no lo necesitan.

Problema general:

Costos en infraestructura elevados y caídas de la plataforma por tener una arquitectura monolítica.

Problemas específicos:

- Desperdicio de recursos tecnológicos porque el escalamiento se realizaba a todo el sistema donde había partes que no necesitaban ese nivel de cómputo.
- Ejecución manual para el escalamiento de la infraestructura.
- Al tener un único sistema, cualquier caída impactaba a toda la plataforma.

Objetivo general:

Migrar progresivamente el sistema de Fandango a microservicios para administrar y gestionar proactivamente los recursos tecnológicos y económicos que se le da a cada servicio.

Objetivos específicos:

- Reducir los costos de infraestructura en Amazon Web Services en 25%.
- Mantener el tiempo en línea del servicio de venta de entradas en al menos 99.999%
- Migrar las funcionalidades y módulos del monolito a microservicios.
- Diseñar políticas de escalamiento según el tráfico que recibe la plataforma y anticiparnos a picos de peticiones hacia los servidores.

3.4 Proyecto de Solución

El constante crecimiento en demanda de la plataforma llevó al equipo a rediseñar la arquitectura de manera distribuida para poder gestionar cada servicio independientemente, como si cada uno fuese un sistema que no conoce que existe el otro. Del mismo modo, poder acoplar y desacoplar servicios sin que afecten el funcionamiento del sistema. Este es un reto mucho más grande de lo que parece, pasas de tener que administrar un solo sistema a tener decenas o centenares de servicios, como también cada uno de estos servicios puede especializarse en un stack en particular que en su conjunto sean las mejores tecnologías para el problema que se quiere realizar. Esto produce que se comience a crear estándares en la construcción de cada uno, nos dimos cuenta de que necesitábamos un plan que también nos permita definir el alcance de cada servicio, hasta donde llega la responsabilidad de cada uno y evitar tener servicios parecidos que vuelva inmantenible el rediseño de la aplicación.

Existía varios retos que se tenía para poder migrar de una arquitectura monolito a microservicios, entre los cuales estaban:

- No se podían parar los desarrollos actuales y pendientes para refactorizar el sistema a microservicios.
- Cada refactorización debía entrar al roadmap, lo cuál hacía que cada uno sea priorizado. El plan de como debíamos atacar este problema era clave, volver a arreglar algo iba a ser muy caro para la compañía.
- La disponibilidad de desarrolladores disponibles y quienes eran estos desarrolladores podía cambiar en el tiempo. Sea por re-priorizaciones del roadmap o por alguna salida o entrada de un nuevo integrante al equipo.
- No solo era desarrollar el microservicio, también se debían considerar los tiempos de rediseños o actualizaciones gráficas, despliegue, programar la conexión del monolito al microservicio.

3.4.1 Alcance

Desacoplar progresivamente los módulos del monolito en microservicios, cada desarrollo de cada microservicio tiene un plazo y recursos propios. Como también se definirá brevemente la arquitectura y funcionalidades de cada microservicio y las razones de su priorización. Los microservicios realizados fueron:

Tabla 2. Alcance microservicios

Microservicio	Inicio	Fin	Desarrolladores
Invoice	20 marzo 2017	23 junio 2017	2
Notification	15 enero 2018	16 marzo 2018	2
Blue	7 mayo del 2018	3 agosto 2018	1
Identity	3 septiembre 2018	30 noviembre 2018	3
Pos	7 enero 2019	22 marzo 2019	4
Voucher	1 abril 2019	5 julio 2019	3

Elaboración: El autor

3.4.2 Limitaciones

- Debido al cierre de operaciones de Fandango Latinoamérica y por políticas de NBC Universal, no ha sido posible gestionar documentos originales para presentar como parte del presente informe.

- El desarrollo de los microservicios se hizo pausado y en algunas ocasiones con espacios amplios entre el desarrollo de cada microservicio por limitaciones en el equipo técnico que debía enfocarse en tareas netamente del negocio.

3.4.3 Metodología

Desacoplamiento

Basándonos en la metodología propuesta por Zhamak Dehghani, donde menciona que una buena estrategia desde donde uno debe partir a dividir un microservicio es utilizando las partes extremas del monolito, lo cuál permitirá al equipo de desarrolladores ir aprendiendo sobre como realizar esta tarea y al equipo de infraestructura a como manipular el despliegue y la operación de estos servicios. Al mencionar parte extrema del monolito, se refiere a esos servicios que no tienen dependencias de otro y que están lejos del core del sistema o pueden coexistir con lo que se tiene actualmente en el monolito.

Se debe evitar que los servicios que se van desacoplando tengan dependencia con el monolito que estamos intentando romper. Lo cual sirve como una técnica para ir definiendo los siguientes microservicios por desarrollar. Por ejemplo, en nuestro caso teníamos el sistema de venta de entradas el cuál, en pocas palabras y para puntualizar en el ejemplo, se tenía un servicio de ticketing que dependía de la conexión con los cines. Lo correcto es primero desacoplar la conexión con los cines y luego ticketing,

porque si fuese el inverso y primero desacoplamos ticketing, este servicio tendría dependencia del monolito para operar porque igual depende de la conexión a los cines para obtener precios, mapa, reservar, comprar y anular. Pero si desacoplamos la conexión a los cines primero, este servicio no necesita del monolito para operar y queda completamente desconectado y el monolito puede usarlo para operar.

Algunos servicios son más difíciles de que al momento de desacoplarlos no tengan dependencia inversa al monolito. Lo que sugiere Zhamak es que el monolito exponga un servicio que permita a este microservicio poder funcionar como si en un futuro su dependencia sea de otro microservicio y no del monolito (Dehgani, 2018). Por ejemplo, si quisiéramos desacoplar el servicio de ticketing antes que el servicio de conexión con los cines se necesitaría que el servicio de conexión con los cines exponga un API con la cuál ticketing pueda interactuar, imaginando que el servicio de conexión con los cines exista. Esto conlleva mucha complejidad porque se necesita tener claro como funcionará este servicio con el que todavía se tendrá dependencia para evitar futuras refactorizaciones, las cuales tomarán tiempo, deberán priorizarse y puede que afecten en los tiempos de despliegue a producción de nuevos microservicios.

Luego de desacoplar los servicios externos, puede que nos veamos en la complejidad de atacar los servicios internos que podrían tener mucha dependencia entre ellos. La manera de poder “despegar” este servicio del resto es refactorizarlo en diferentes partes dependiendo de su dominio. Por ejemplo, si tuvieses una única sesión en la cuál guardas información del usuario y su carrito de compras, lo adecuado es que cada uno se separe en distintas sesiones y así al momento de extraer alguno de estos servicios pueda llevarse la sesión consigo.

El desacoplamiento vertical es otro punto clave en la refactorización de un monolito a microservicios. Donde puede que en el monolito tengamos una arquitectura modelo vista controlador. En este punto podemos atacar el desacoplamiento de dos formas, una es simplemente abstrayendo la capa lógica de la capa visual o, probablemente de la más recomendada, es que cada microservicio exponga su propia capa visual. Por ejemplo, el servicio de autenticación y registro de usuarios es probable que quisieras que no solo cuente con la lógica para crear o logear usuarios, sino también que también cuente con su interfaz gráfica. De esta forma, no importa de donde lo llames, sea de un App, Web o Quiosco, puedas invocar el mismo servicio para autenticar o registrar usuarios. Esto también te permite centralizar cada funcionalidad en cada servicio y tener el control sobre lo que sucede, como también genera independencia para actualizaciones en el futuro y no esperar que quienes lo usen deban adaptarse al servicio.

El desacoplamiento vertical también contempla el desacoplamiento de la base de datos. Es importante no caer en el anti-patrón de solo separar la capa lógica, pero seguir teniendo la misma base de datos compartida entre todos los microservicios. Recordemos que una de las principales razones por las que empezamos el desacoplamiento era porque la base de datos se encontraba utilizando gran cantidad de sus recursos. No considerar eso haría que toda la refactorización nos lleve al mismo problema inicial.

Una estrategia para priorizar servicios por desacoplar es identificar esos servicios del sistema que suelen tener más cambios y negociar con las necesidades del negocio para que en un futuro cambio se pueda ampliar el tiempo de desarrollo para poder programar el microservicio para esa funcionalidad.

Se puede partir de lo macro a lo micro, esto quiere decir que podemos empezar refactorizando el código interno del monolito, separándolo por

dominio, para luego cada dominio ir desacoplándolo en un microservicio. Esto es muy recomendable cuando tenemos un monolito demasiado acoplado y desordenado, sin patrones ni principios de programación.

Como metodología para organizar el proyecto y equipo se utilizó Scrum. Scrum es “un marco de trabajo ligero que permite a las personas, equipos y organizaciones generar valor a través de soluciones adaptables para problemas complejos” (Scrum Org, 2021).

Scrum

Scrum es un proceso cíclico de desarrollo de software, el cuál permite tener entregables tangibles que van definiendo y validando si se cumplen las necesidades del negocio. El cuál cuenta con artefactos como Product Backlog y Sprint Backlog (Garcia Fuentes, 2019).

En la metodología Scrum se cuenta con distintos roles (Hema, y otros, 2020):

- Scrum Master: Asegura el cumplimiento de las ceremonias de Scrum. Es un facilitador.
- Product Owner: Stakeholder o representante del cliente.
- Scrum Team: El equipo que va a cumplir con las tareas.

Scrum también está compuesta de eventos o ceremonias (Schwaber & Sutherland, 2021):

- Sprint, es el corazón de scrum y es el que define el tiempo en el que se va a trabajar en un conjunto de tareas. Una vez seteado el tiempo, este no puede pararse salvo que el Product Owner lo decida.
- Sprint Planning, Es el planeamiento del Sprint, las tareas que se verán durante ese periodo y se construye en conjunto con el Scrum Team.
- Daily Scrum, es una reunión de 15 minutos, donde se busca mapear el avance del sprint e identificar problemas. En esta ceremonia deben

resolverse 3 preguntas por cada miembro, ¿qué hiciste ayer?, ¿qué harás hoy?, ¿has tenido o tienes algún impedimento?

- Sprint Review, se realiza al final del sprint y se inspecciona el incremento que ha tenido el producto y se adapta el product backlog si es necesario. En esta ceremonia uno de los puntos más importantes es haber cumplido con el *definition of done* en cada tarea. El *definition of done* es lo que la organización requiere de una tarea antes de ser entragada al cliente final, básicamente que es lo que define que una tarea está completamente terminada (Scrum Inc, 2021).
- Sprint Retrospective, es una oportunidad para el Scrum Team para reflexionar sobre lo vivido en los sprints, esta ceremonia sucede una vez al mes y busca identificar problemas, dolores o incomodidades dentro del equipo.

Scrum se puede definir en un solo gráfico, donde se muestra el proceso donde se crean todas las tareas en un Product Backlog, luego esas tareas van entrando a los sprints, los cuales son cíclicos y cada final de sprint debe haber un incremental al producto final.

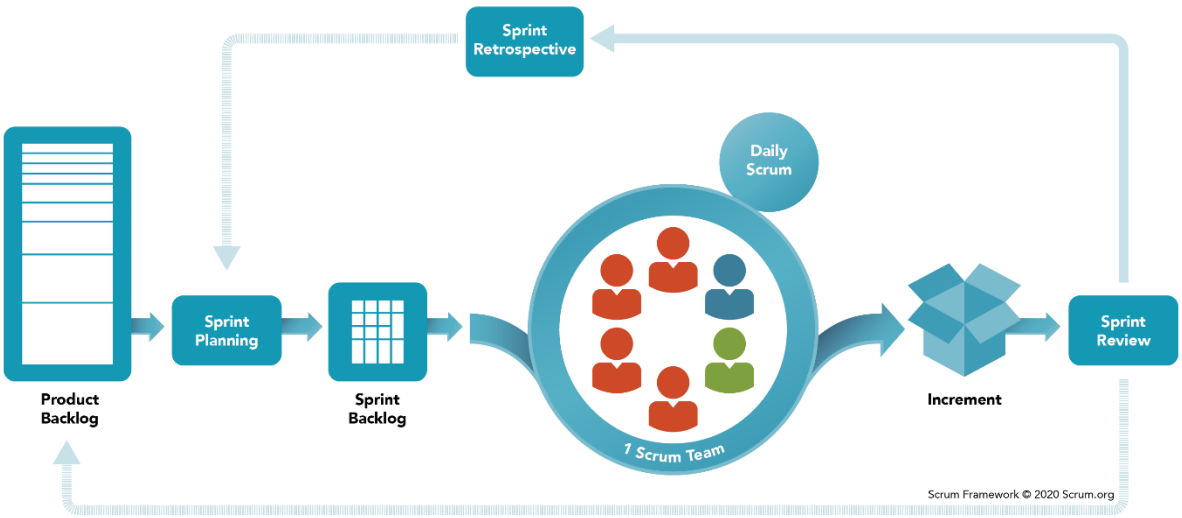

Figura 10. The Scrum Framework

Elaboración: scrum.org

Test-Driven Development

Como metodología de desarrollo se utilizó TDD, *Test-driven development*, la cual es una técnica de programación donde las pruebas son programadas antes que el código fuente (Siniaalto & Abrahamsson, 2007). El uso de TDD impacta positivamente en la forma en la que escribimos código, debido a que el diseño de la aplicación es más sencilla ya que te fuerza a desacoplar el código, asimismo, se disminuya la cantidad de errores que pueden llegar a producción debido a que se vuelve predecible el impacto que tiene cada cambio, se calcula una posible mejora del 21% en la calidad del código cuando se utiliza TDD (Khanam & Ahsan, 2017).

Escalamiento predictivo

Según Amazon Web Services es posible hacer escalamiento predictivo (Amazon Web Services, 2021) en los siguientes casos:

- Tráfico cíclico, cuando tienes una gran cantidad de uso de recursos en la misma franja horaria.
- Recurrentes procesos masivos como procesamiento de batches, pruebas o análisis de data (reportes).
- Aplicaciones que toman mucho tiempo en ser inicializadas causando un gran impacto en la latencia.

El auto-escalamiento previene el sobre-aprovisionamiento de recursos, permitiendo ahorrar costos y utilizando los recursos al máximo. Existiendo principalmente dos técnicas, una basada en el uso de reglas que definen las condiciones para escalar y la otra se basa en las tendencias de una serie de análisis, teoría de control y aprendizaje, donde te adelantas a la necesidad de escalar, lo cuál es mucho más poderoso porque escalar cuando lo necesitas puede que, dependiendo del tiempo que demore tu software en

arrancar, tome mucho tiempo y la demanda sobrepase la capacidad de tu infraestructura, generando una caída (Biswas, Mahumdar, Nandy, & El-Haraki, 2017).

Para hacer un escalamiento predictivo se tiene tres opciones (Gimenez, 2020):

- **Proyección de carga:** Analiza el histórico de los últimos 14 días y proyecta la demanda para los siguientes 2 días. Se actualiza diariamente.
- **Acciones de escalamiento programadas:** Agrega o elimina recursos de acuerdo con las proyecciones. Pero el escalamiento varía según la hora del día.
- **Comportamiento de capacidad máxima:** Se designa una capacidad mínima y máxima para cada recurso y se deja que el mismo AWS defina el uso de recursos automáticamente. Tu solo controlas si tu aplicación en algún momento necesitará más capacidad de la máxima programada.

3.4.4 Etapas

Un punto importante para el éxito de estos proyectos es la atomización de los pasos que se van a seguir para el futuro del desacoplamiento del monolito en microservicios. Esto es clave porque al ser un proyecto largo, probablemente de años, es posible que las prioridades del negocio lleven en muchos momentos a suspender el avance del proyecto, o porque el negocio pivoteó a otra idea o algo más común es que el líder que estaba empujando que el proyecto avanza se va de la empresa y la nueva persona no le da su soporte a este proyecto.

Considerando lo anterior, las etapas de este proyecto se basaban en priorizar los módulos según la metodología presentada. Esta priorización se iba a intentar distribuir a lo largo del año en un roadmap que se comparte con las necesidades del negocio, teniendo claro que no iba a ser posible que se realice todo a la vez, esta

priorización se enfocó en dos puntos, aprender a hacer desacoplamientos y aislar los servicios críticos del sistema que en principio no tengan dependencia inversa con el monolito, con esto podríamos proteger la base de datos de sobrecargas y ganar tiempo para seguir avanzando con el resto de monolitos. Era claro que el tráfico seguiría creciendo porque nuevos cines se iban integrando a la plataforma, por lo cuál el punto más importante del negocio que era vender entradas debía siempre estar disponible, más adelante explicaré como sucedió ese desacoplamiento.

1. Aprender a desacoplar con servicios sencillos de operar y que no afecten al core del sistema.

2. Servicios críticos del sistema que no tengan dependencia inversa al monolito.

3. Servicios que permitan seguir operando el negocio, aunque estos no estén disponibles.

4. Eliminar todos los servicios de business intelligence que se conectan con la base de datos, sea producción o réplica.

5. Refactorización del monolito para que todo esté ordenado por dominios.

6. Desacoplamiento de servicios que saturen la base de datos de conexiones o CPU.

7. El resto de los servicios serán priorizados según su uso por los clientes o cantidad de modificaciones que tengan.

3.4.5 Solución

1. **Invoice**
   Necesidades

   Se necesita generar la factura electrónica en cada país en los que Fandango Latinoamérica tenga presencia y esté obligado a facturar localmente. Como también generar notas de crédito y la anulación de las facturas generadas. Esto porque se tiene que cumplir la regulación local.

   El sistema actual al estar dentro del monolito generaba algunos problemas de inestabilidad debido a que el proceso para generar la factura electrónica era lento y eso aumentaba la latencia en la respuesta del servidor. Adicionalmente, como ningún módulo dependía directamente del facturador electrónico, iba a ser el módulo más sencillo para desacoplar.

   Stakeholders

Tabla 3. Stakeholders del microservicio Invoice

Entidad/Persona	**Descripción**
Contabilidad	El área de contabilidad de Fandango necesitaba que se facture automáticamente y tener reportes.
SUNAT	Proveedor de facturación electrónica en Perú y centro américa (se emitía la boleta desde Perú)
FyM	Proveedor de facturación electrónica en Colombia
Bsale	Proveedor de facturación electrónica en Chile
Stupendo	Proveedor de facturación electrónica en Ecuador
GlobalSis	Proveedor de facturación electrónica en Argentina
Fandango	El sistema de Fandango consumirá de Invoice para la generación de facturas electrónicas vía API.
Infraestructura	Administrar el servicio en la nube.

Elaboración: El autor

Requisitos

- Generar facturas electrónicas con el desagregado de impuestos correspondientes por cada venta realizada.
- Poder generar una nota de crédito por la anulación de una factura electrónica.
- Poder anular las facturas electrónicas.
- Generar reportes de las facturas emitidas, notas de crédito emitidas y anulaciones realizadas, en un rango de fechas según necesite el cliente.
- Poder visualizar las facturas electrónicas y sus respectivos lotes.
- Las facturas electrónicas tienen un plazo máximo de 30 días para generarse una vez realizada la venta.
- En Bolivia generar un documento con las características necesarias de acuerdo con la ley. Debe tener un número correlativo y tener la información del a empresa.
- Debe generar una URL única para cada factura emitida, la cuál se le entregará al sistema que consuma Invoice y este será el encargado de distribuirla.
- Cada venta que se realice debe ejecutar este servicio asíncronamente y asegurarse que se reintente en caso de error en comunicación.
- Se debe refactorizar la forma en al que se genera la factura electrónica para que utilice este nuevo servicio.

Matriz de responsabilidades

En el siguiente cuadro, se menciona a cada miembro del proyecto, donde a su vez se puede visualizar el rol que tuvo en el proyecto para este microservicio como también el cargo que tiene en la compañía.

Tabla 4. Matriz de responsabilidades de Invoice

Responsable	**Rol en el proyecto**	**Cargo**
Jhon Campos	Desarrollador / Gestor del proyecto	Desarrollador
Sebastián Burgos	Desarrollador	Desarrollador
Damaris Lozano	Responsable / Contacto con proveedores	Contadora

Elaboración: El autor

Product Backlog

Tabla 5. Product Backlog de Invoice

ID	**Descripción**	**Story Points**
I01	[API] Autenticación	8
I02	[API] CRUD Factura	13
I03	[API] Crear Nota de Crédito	8
I04	[API] Anular Factura	8
I05	Login	3
I06	Registro de usuarios	3
I07	Vista de facturas + Filtros	3
I08	Vista de Lotes + Filtros	3
I09	Reporte de Facturas por país y rango de fecha	5
I10	[SUNAT] enviar XML para generar boleta electrónica	3
I11	[SUNAT] enviar XML para generar nota de crédito	3
I12	[SUNAT] Confirmación de boleta generada	3
I13	[SUNAT] Rescatar boleta generada	5
I14	[FyM] JSON para generar factura electrónica	5
I15	[FyM] Anular factura electrónica	3
I16	[FyM] Obtener boleta generada	5
I17	[BSale] JSON para generar factura electrónica	3

I18	[Bsale] Anular orden	5
I19	[BSale] Obtener factura generada	5
I20	[Stupendo] JSON para generar factura electrónica	5
I21	[Stupendo] Anular factura electrónica	3
I22	[Stupendo] Obtener factura generada	5
I23	[GlobalSis] XML para generar factura electrónica	5
I24	[GlobalSis] Anular factura generada	3
I25	[GlobalSis] Obtener factura generada	5
I26	Generar factura computarizada	13
I27	Cola para generar factura	13
I28	Generación de lotes para enviar a proveedor	8
I29	Refactorización de la generación de facturas electrónicas de Fandango al nuevo servicio	13

Elaboración: El autor

Sprint Backlog

Tabla 6. Sprint 1 de Invoice

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsable**
Análisis	Diseño y creación del modelo de base de datos.	Jhon Campos
Análisis	Diseño y creación de la arquitectura del proyecto	Jhon Campos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Jhon Campos

Elaboración: El autor

Tabla 7. Sprint 2 de Invoice

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsable**
Análisis	Diseño de las tramas de los servicios	Jhon Campos
TDD	Programación de las pruebas unitarias de I01	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I02	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I03	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I04	Jhon Campos
TDD	Programación de las pruebas unitarias de I05	Jhon Campos

Elaboración: El autor

Tabla 8. Sprint 3 de Invoice

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de I06	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I07	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I08	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I09	Jhon Campos
TDD	Programación de las pruebas unitarias de las tramas	Jhon Campos

Elaboración: El autor

Tabla 9. Sprint 4 de Invoice

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de I10, I11, I12, I13	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I14, I15, I16	Sebastián Burgos

TDD	Programación de las pruebas unitarias de I17, I18, I19	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I20, I21, I22	Jhon Campos
TDD	Programación de las pruebas unitarias de I23, I24, I25	Jhon Campos
TDD	Programación de las pruebas unitarias de I26	Jhon Campos

Elaboración: El autor

Tabla 10. Sprint 5 de Invoice

Sprint 5		
Tipo de Tarea	**Descripción**	**Story Points**
Implementación	Programación de la funcionalidad I01	Sebastián Burgos
Implementación	Programación de la funcionalidad I02	Jhon Campos
Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 11. Sprint 6 de Invoice

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsable**
Implementación	Programación de la funcionalidad I03	Sebastián Burgos
Implementación	Programación de la funcionalidad I04	Sebastián Burgos
Implementación	Programación de la funcionalidad I05	Jhon Campos
Implementación	Programación de la funcionalidad I06	Jhon Campos

Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 12. Sprint 7 de Invoice

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I07	Sebastián Burgos
Implementación	Programación de la funcionalidad I08	Jhon Campos
Implementación	Programación de la funcionalidad I09	Jhon Campos
Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 13. Sprint 8 de Invoice

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I10, I11, I12, I13	Jhon Campos

Implementación	Programación de la funcionalidad I14, I15, I16	Sebastián Burgos
Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 14. Sprint 9 de Invoice

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I17, I18, I19	Sebastián Burgos
Implementación	Programación de la funcionalidad I20, I21, I22	Jhon Campos
Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 15. Sprint 10 de Invoice

Sprint 10		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I23, I24, I25	Sebastián Burgos
Implementación	Programación de la funcionalidad I26	Jhon Campos

Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 16. Sprint 11 de Invoice

Sprint 11		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I27, I28	Sebastián Burgos
Pruebas	Verificación de que las pruebas unitarias funcionen	Jhon Campos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 17. Sprint 12 de Invoice

Sprint 12		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea I29	Sebastián Burgos
Despliegue	Pase al ambiente de pruebas en la nube	Jhon Campos

Elaboración: El autor

Tabla 18. Sprint 13 de Invoice

Sprint 13		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a producción	Jhon Campos
Pruebas	Pruebas en producción	Sebastián Burgos

Cronograma

El proyecto contó con 13 *sprints*, los cuales empezaron el 20 de marzo del 2017 y finalizaron 23 de junio del 2017

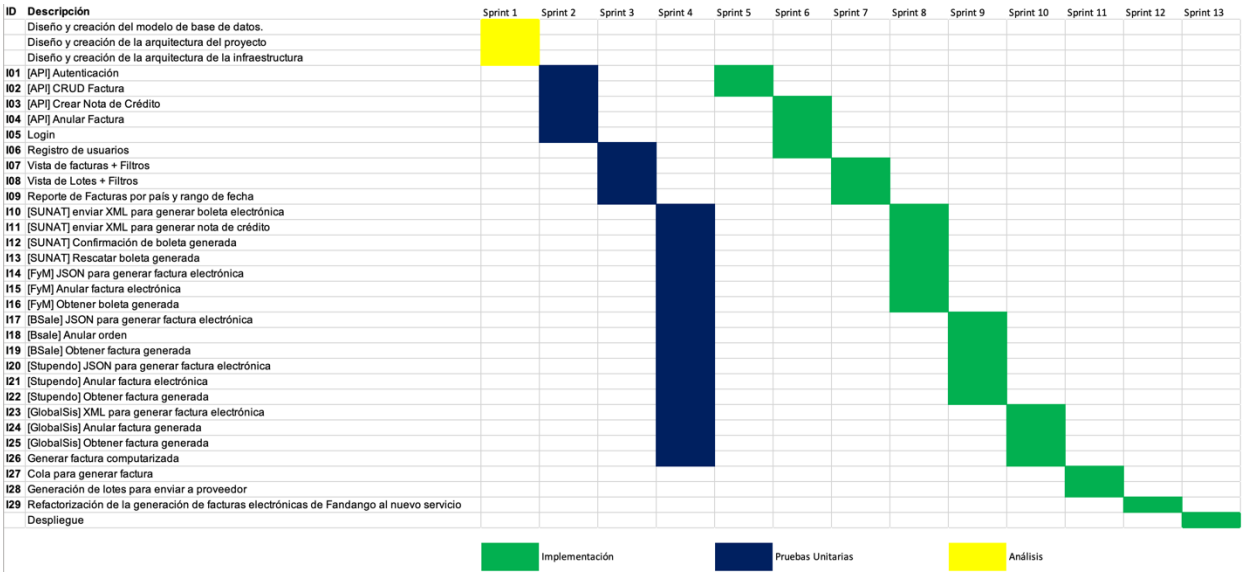

Figura 11. Cronograma Invoice

Elaboración: El autor

Arquitectura

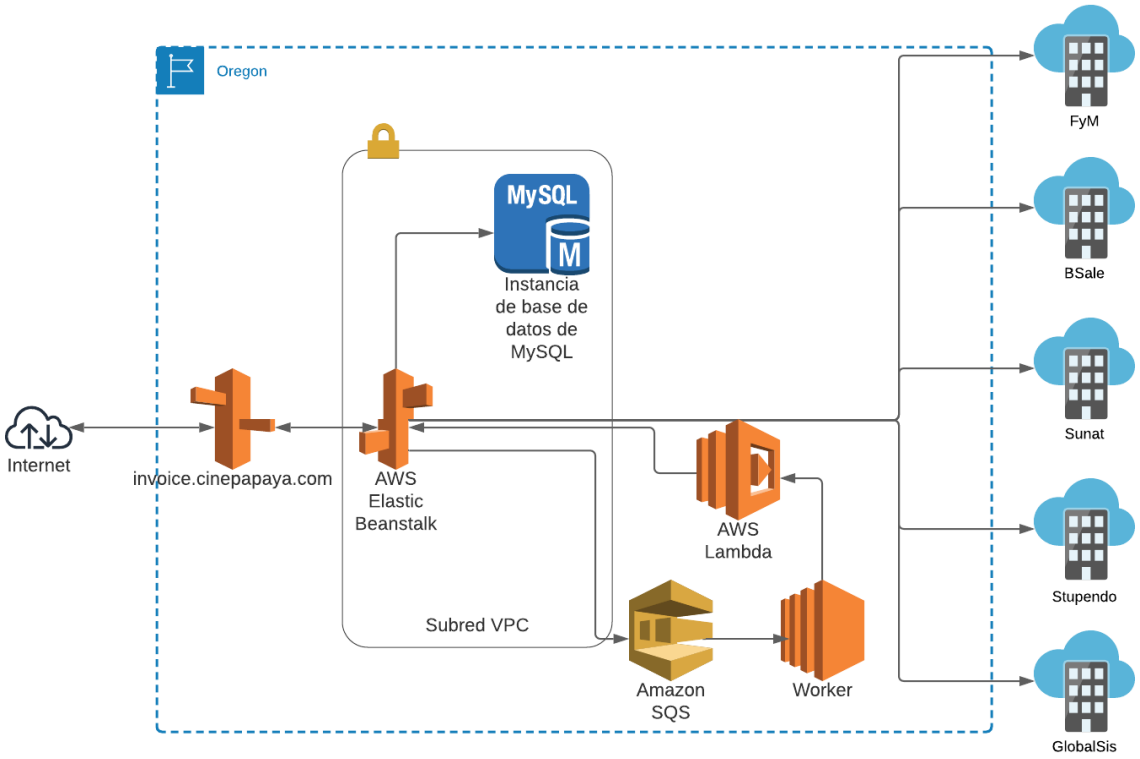

Figura 12. Arquitectura de Invoice

Elaboración: El autor

Costos

Tabla 19. Costo mensual promedio de AWS en Invoice

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
AWS Lambda	Cantidad de solicitudes (5000000)	USD$ 0.80
Amazon EC2	Sistema operativo (Linux), Cantidad (1), Pricing strategy (EC2 Instance Savings Plans 1 año Sin pagos iniciales), Cantidad de almacenamiento (30 GB), Tipo de instancia (t3.medium)	USD$ 22.05
Elastic Load Balancing	Número de balanceadores de carga de aplicaciones (1)	USD$ 89.43
Amazon Simple Queue Service (SQS)	Solicitudes de cola estándar (1 millones por mes)	USD$ 0.40
Amazon EC2	Sistema operativo (Linux), Cantidad (1), Pricing strategy (EC2 Instance Savings Plans 1 año Sin pagos iniciales), Cantidad de almacenamiento (30 GB), Tipo de instancia (m3.X large)	USD$ 141.70
Amazon Route 53	Zonas alojadas (1)	USD$ 2.50
Amazon RDS for MySQL	Almacenamiento para cada instancia RDS (SSD de uso general (gp2)), Cantidad de almacenamiento (200 GB), Cantidad (1), Tipo de instancia (db.t2 Medium), Opción de implementación (Multi-AZ), Modelo de precios (OnDemand)	USD$ 145.28
	Total	**USD$ 402.16**

Elaboración: El autor

2. Notification

Necesidades

Durante el desarrollo del monolito en Fandango se hicieron distintas formas para enviar notificaciones a los usuarios, estas fueron Email, SMS y Push Notification. Se necesita que esto esté centralizado en un solo servicio que sea responsable de asegurarse de que se envíe la notificación según el medio indicado.

Stakeholders

Tabla 20. Stakeholders de Notification

Entidad/Persona	**Descripción**
Marketing	Necesitan que sus campañas y la comunicación con el usuario se ejecute correctamente usando distintos medios
Fandango	Necesita que las notificaciones que le hace al usuario sean por la venta de una entrada, cambio de contraseña, nuevo estreno, etc. Le llegue por email o push notification
Identity	Necesita que se le envíe un número aleatorio al usuario para validar que el teléfono y correo que está registrando le pertenecen.
Mandill (Mailchimp)	Proveedor de envío de emails transaccionales
Infobip	Proveedor de envío de SMS
Clevertap	Proveedor de envío de Push Notifications
Infraestructura	Administrar el servicio en la nube.

Elaboración: El autor

Requisitos

- Adecuar el sistema de Fandango para que todas las notificaciones que se envían sean por medio de este nuevo servicio.
- Asegurarse de que la notificación sea enviada al destinatario lo más pronto posible utilizando a los mejores proveedores del mercado.
- Poder obtener la información del estado de una notificación.
- Enviar notificación según el medio que el cliente (Fandango) decida, los cuales pueden ser SMS, Email o Push Notification.
- Se debe actualizar todos los lugares donde hoy se envía notificaciones para que utilicen el nuevo servicio.

Matriz de Responsabilidades

Tabla 21. Matriz de Responsabilidades de Notification

Responsable	**Rol en el Proyecto**	**Cargo**
Sebastián Burgos	Líder del proyecto / Desarrollador	Líder del equipo de integraciones
Juana Custodio	Desarrollador	Desarrollador

Elaboración: El autor

Product Backlog

Tabla 22. Product Backlog de Notification

ID	**Descripción**	**Story Points**
N01	CRUD Notificación	13
N02	Enviar Notificación	13
N03	[Email – Mandrill] Enviar cuerpo del email, asunto y receptor	8
N04	[SMS – Infobip] Enviar mensaje, número de teléfono y país	8

N05	[Push Notification – Clevertap] Enviar Profile ID y mensaje	13
N06	Cola de reintento de notificación	13
N07	Refactorizar envío de notificación de Fandango	26

Elaboración: El autor

Sprint Backlog

Tabla 23. Sprint 1 de Notification

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño y creación del modelo de base de datos.	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura del proyecto	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Sebastián Burgos

Elaboración: El autor

Tabla 24. Sprint 2 de Notification

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño de las tramas de los servicios	Sebastián Burgos
TDD	Programación de las pruebas unitarias de N01	Sebastián Burgos
TDD	Programación de las pruebas unitarias de N02	Juana Custodio
TDD	Programación de las pruebas unitarias de N06	Juana Custodio

Elaboración: El autor

Tabla 25. Sprint 3 de Notification

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de N03	Sebastián Burgos
TDD	Programación de las pruebas unitarias de N04	Juana Custodio
TDD	Programación de las pruebas unitarias de N05	Juana Custodio

Elaboración: El autor

Tabla 26. Sprint 4 de Notification

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad N01	Juana Custodio
Implementación	Programación de la funcionalidad N02	Sebastián Burgos
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Juana Custodio
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 27. Sprint 5 de Notification

Sprint 5		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad N03	Sebastián Burgos
Implementación	Programación de la funcionalidad N04	Juana Custodio
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos

Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Juana Custodio
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 28. Sprint 6 de Notification

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad N05	Sebastián Burgos
Implementación	Programación de la funcionalidad N06	Juana Custodio
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Juana Custodio
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 29. Sprint 7 de Notification

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a Producción	Sebastián Burgos
Pruebas	Pruebas en producción	Sebastián Burgos

Elaboración: El autor

Tabla 30. Sprint 8 de Notification

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea N07 (tomará 2 sprints)	Juana Custodio

Pruebas	Verificar que las notificaciones se estén enviando	Sebastián Burgos

Elaboración: El autor

Tabla 31. Sprint 9 de Notification

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsable**
Refactorización	Finalización del desarrollo de la tarea N07	Juana Custodio
Pruebas	Verificar que las notificaciones se estén enviando	Sebastián Burgos

Elaboración: El autor

Cronograma

El proyecto contó con 9 *sprints*, los cuales empezaron el 15 de enero del 2018 y finalizaron 16 de marzo del 2018

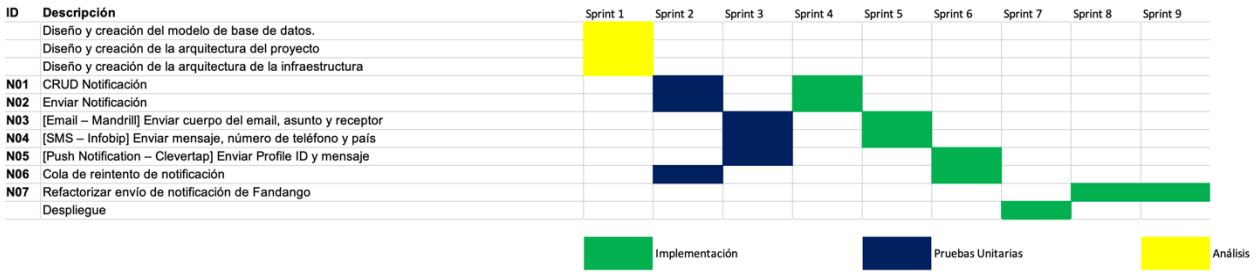

Figura 13. Cronograma Notification

Elaboración: El autor

Arquitectura

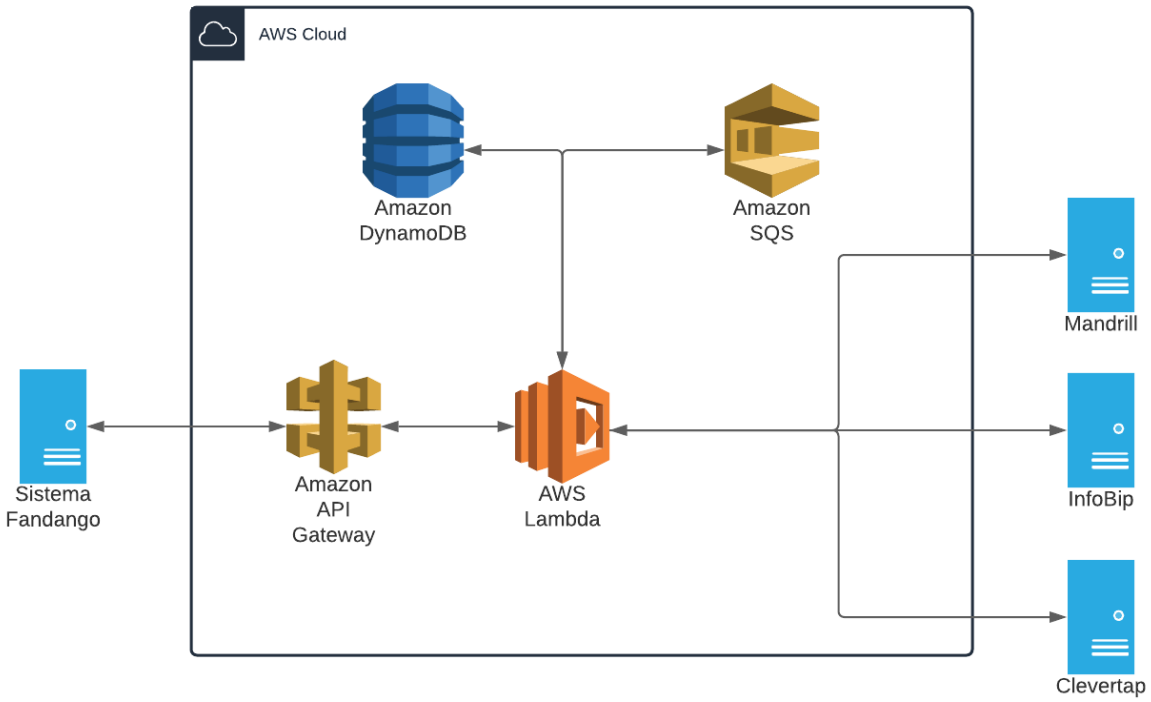

Figura 14. Arquitectura de Notification

Elaboración: El autor

Costos

Tabla 32. Costo mensual promedio de AWS de Notification

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
AWS Lambda	Cantidad de solicitudes (100000000)	USD$ 33.97
Amazon Simple Queue Service (SQS)	Solicitudes de cola estándar (100 millones por mes)	USD$ 40.00
Amazon DynamoDB	Tamaño promedio del elemento (todos los atributos) (1 KB), Escribir el plazo de la capacidad reservada (1 año), Leer el plazo de la capacidad reservada (1 año), Tamaño del almacenamiento de datos (200 GB)	USD$ 76.14
Amazon API Gateway	Unidades de solicitudes de la API HTTP (millones), Tamaño promedio de cada solicitud (34 KB), Unidades de solicitud de la API REST (millones), Tamaño de memoria caché (GB) (Ninguno), Unidades de mensaje WebSocket (miles), Tamaño promedio del mensaje (32 KB), Solicitudes (por mes), Solicitudes (100 por mes)	USD$ 350.00
	Total	**USD$ 500.11**

Elaboración: El autor

3. Blue

Necesidades

Sé necesitaba flexibilidad en el manejo de las reglas antifraude porque en el monolito cada modificación de reglas significaba un cambio en el código y se requería que el área de riesgos pueda tener el control inmediato de las modificaciones para correr experimentos. Asimismo, este era un servicio con alta concurrencia pero que no era indispensable para la venta de entradas al cine.

Stakeholders

Tabla 33. Stakeholders de Blue

Entidad/Persona	**Descripción**
Fandango	Sistema que consumirá el API para validar si una transacción es fraudulenta
Riesgos	Área de Fandango encargada del mantenimiento de reglas antifraude.
InAuth	Proveedor que genera un fingerprint de cada dispositivo que utilizan los usuarios.
Infraestructura	Administración del servicio en la nube.

Elaboración: El autor

Requisitos

- Se debe generar reportes de los experimentos que genere el área de riesgos.
- El sistema debe ser capaz de generar un árbol de decisiones administrable por el área de riesgos.
- El sistema debe responder en segundos cada verificación de fraude.

- El sistema deba almacenar los datos de cada verificación como data para futuros análisis.
- El sistema debe soportar una alta concurrencia de peticiones.
- Fandango debe utilizar este nuevo servicio para la validación antifraude.

Matriz de Responsabilidades

Tabla 34. Matriz de Responsabilidades de Blue

Responsable	**Rol en el proyecto**	**Cargo**
Sebastián Burgos	Líder del proyecto / Desarrollador	Líder del equipo de integraciones
Miguel Mini	Desarrollador	Desarrollador
Andrea Allen	Cliente y tester	Analista de Riesgos

Elaboración: El autor

Product Backlog

Tabla 35. Product Backlog de Blue

ID	**Descripción**	**Story Points**
B01	Crear orden	3
B02	Actualizar orden	3
B03	Crear árbol de decisión	13
B04	[InAuth] Generar fingerprint con el SDK	5
B05	Módulo de decisión por similitud	13
B06	Módulo de decisión por lista blanca o negra	8
B07	Módulo de decisión por velocidad - repetición de transacciones en X tiempo	13
B08	Módulo de decisión por velocidad – repetición de fingerprint en C tiempo	13

B09	Módulo de decisión por patrón	13
B10	Reporte de reglas más testeadas	8
B11	Reporte de experimentos	13
B12	Login	5
B13	Registro	5
B14	Interfaz para administrar el árbol de decisiones	13
B15	Animaciones para administrar el árbol de decisiones	13
B16	Panel para crear reglas	13
B17	Refactorización en Fandango para que utilice este nuevo servicio	13

Elaboración: El autor

Sprint Backlog

Tabla 36. Sprint 1 de Blue

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño y creación del modelo de base de datos.	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura del proyecto	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Sebastián Burgos

Elaboración: El autor

Tabla 37. Sprint 2 de Blue

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño de las tramas de los servicios	Sebastián Burgos

TDD	Programación de las pruebas unitarias de B01	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B02	Miguel Miní
TDD	Programación de las pruebas unitarias de B04	Miguel Miní

Elaboración: El autor

Tabla 38. Sprint 3 de Blue

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de B03	Miguel Miní
TDD	Programación de las pruebas unitarias de B05	Sebastián Burgos

Elaboración: El autor

Tabla 39. Sprint 4 de Blue

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de B06	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B07	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B08	Miguel Miní
TDD	Programación de las pruebas unitarias de B09	Miguel Miní

Elaboración: El autor

Tabla 40. Sprint 5 de Blue

Sprint 5		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de B10	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B11	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B12	Miguel Miní
TDD	Programación de las pruebas unitarias de B13	Miguel Miní
TDD	Programación de las pruebas unitarias de B14	Miguel Miní

Elaboración: El autor

Tabla 41. Sprint 6 de Blue

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de B15	Sebastián Burgos
TDD	Programación de las pruebas unitarias de B16	Miguel Miní
Implementación	Programación de la funcionalidad B01	Sebastián Burgos
Implementación	Programación de la funcionalidad B02	Miguel Miní
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 42. Sprint 7 de Blue

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad B03	Sebastián Burgos
Implementación	Programación de la funcionalidad B04	Miguel Miní
Implementación	Programación de la funcionalidad B05	Miguel Miní
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 43. Sprint 8 de Blue

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad B06	Sebastián Burgos
Implementación	Programación de la funcionalidad B07	Miguel Miní
Implementación	Programación de la funcionalidad B08	Sebastián Burgos
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 44. Sprint 9 de Blue

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad B09	Sebastián Burgos
Implementación	Programación de la funcionalidad B10	Sebastián Burgos
Implementación	Programación de la funcionalidad B11	Miguel Miní
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 45. Sprint 10 de Blue

Sprint 10		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad B12	Miguel Miní
Implementación	Programación de la funcionalidad B13	Sebastián Burgos
Implementación	Programación de la funcionalidad B14	Miguel Miní
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 46. Sprint 11 de Blue

Sprint 11		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad B15	Sebastián Burgos

Implementación	Programación de la funcionalidad B16	Miguel Miní
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Miguel Miní
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 47. Sprint 12 de Blue

Sprint 12		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a Producción	Sebastián Burgos
Pruebas	Pruebas en producción	Sebastián Burgos

Elaboración: El autor

Tabla 48. Sprint 13 de Blue

Sprint 13		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea B17	Miguel Miní
Pruebas	Verificar validaciones antifraude	Sebastián Burgos

Elaboración: El autor

Cronograma

El proyecto contó con 9 *sprints*, los cuales empezaron el 7 de mayo del 2018 y finalizaron 3 de agosto del 2018

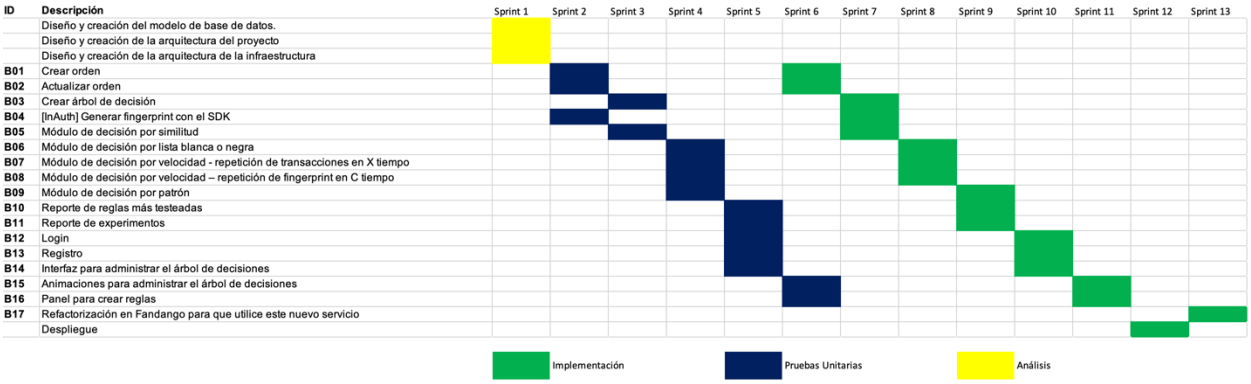

Figura 15. Cronograma Blue

Elaboración: El autor

Arquitectura

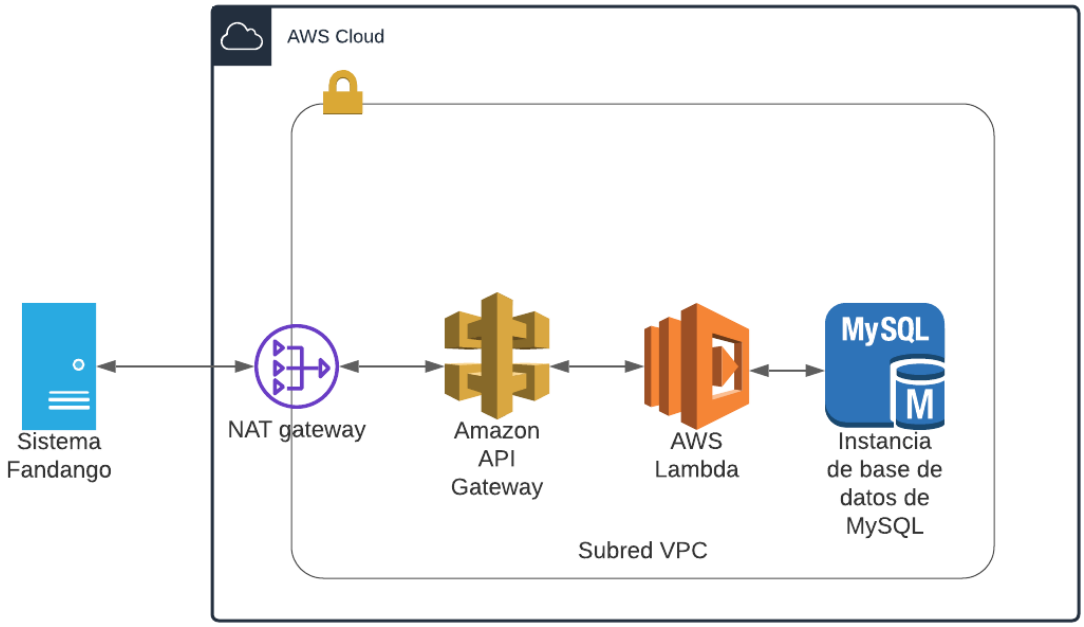

Figura 16. Arquitectura de Blue

Elaboración: El autor

Costos

Tabla 49. Costo mensual promedio de AWS de Blue

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
AWS Lambda	Cantidad de solicitudes (3000000)	USD$ 0.40
Amazon Aurora MySQL-Compatible	Registros de cambios por instrucción (0.38), Nodos (1), Tipo de instancia (db.r3.large), Familia de instancias (Memory optimized), Modelo de precios (OnDemand)	USD$ 211.70
Amazon API Gateway	Unidades de solicitudes de la API HTTP (millones), Tamaño promedio de cada solicitud (34 KB), Unidades de solicitud de la API REST (millones), Tamaño de memoria caché (GB) (Ninguno), Unidades de mensaje WebSocket (miles), Tamaño promedio del mensaje (32 KB), Solicitudes (3 por mes), Solicitudes (3 por mes)	USD$ 13.50
	Total	**USD$ 225.60**

Elaboración: El autor

4. Identity

Necesidades

Uno de los servicios que más ataques de fuerza bruta recibía, sin éxito, pero que igual saturaban mucho nuestros servidores. Sobretodo, en momento pico era un punto que más dolor causaba a la plataforma. Dicho esto, se podía comprar entradas al cine sin tener una cuenta, por lo cuál este servicio era indispensable y queríamos poder apagarlo en momentos críticos para centrar nuestros esfuerzos en la venta de entradas al cine. Debía estar fuera del monolito.

Stakeholders

Tabla 50. Stakeholders de Identity

Entidad/Persona	**Descripción**
Fandango	Sistema que se conectará a Identity para identificar usuarios
Marketing	Requiere conocer a los usuarios que utilizan la plataforma
Riesgos	Requiere identificar correctamente a cada usuario
Infraestructura	Poder administrar el servicio.

Elaboración: El autor

Requisitos

- Mantener la sesión del usuario sin importar el servicio que esté utilizando a través de todo Fandango.
- Utilizar *invisible captcha* para evitar que robots intenten cientos de autenticaciones por segundo.
- Implementar un rate limiter para evitar que robots intenten cientos de autenticaciones por segundo.
- Debe poder usarse tanto en la Web como en dispositivos móviles.
- Debe cuidar la información de los usuarios, debe solo viajar la información mínima y si se desea más datos del usuario no deben

consultarse del lado del cliente, sino de servidor a servidor y de forma encriptada.

- Refactorizar la Web de Fandango para que utilice este nuevo sistema de autenticación y registro de usuarios
- Refactorizar las Apps de Android y iOS para que utilicen este nuevo servicio.

Matriz de Responsabilidades

Tabla 51. Matriz de Responsabilidades de Identity

Responsable	**Rol en el proyecto**	**Cargo**
Cristian Llanos	Líder del proyecto / Desarrollador	Software Architect
Gian Carlos Huachin	Desarrollador	Líder del equipo de servicios
Sebastián Burgos	CTO	CTO
Giomar Rodriguez	Desarrollador Android y iOS	Líder del equipo de móviles

Elaboración: El autor

Product Backlog

Tabla 52. Product Backlog de Identity

ID	**Descripción**	**Story Points**
I01	CRUD usuarios	13
I02	Login	13
I03	[API] Consultar información de usuario	8
I04	[Cookie] ID, nombre y correo de usuario en JWT	8
I05	Login con redes sociales	13
I06	Compatibilidad de la sesión con dispositivos móviles	13

I07	Compatibilidad de vistas con dispositivos móviles	13
I08	Refactorización de la autenticación y registro de la Web	13
I09	Refactorización de la autenticación y registro del App en Android	13
I10	Refactorización de la autenticación y registro del App en iOS	13

Elaboración: El autor

Sprint Backlog

Tabla 53. Sprint 1 de Identity

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño y creación del modelo de base de datos.	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura del proyecto	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Sebastián Burgos

Elaboración: El autor

Tabla 54. Sprint 2 de Identity

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño de las tramas de los servicios	Sebastián Burgos
TDD	Programación de las pruebas unitarias de I01	Cristian Llanos
TDD	Programación de las pruebas unitarias de I02	Gian Carlos Huachín

Elaboración: El autor

Tabla 55. Sprint 3 de Identity

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de I03	Cristian Llanos
TDD	Programación de las pruebas unitarias de I04	Cristian Llanos
TDD	Programación de las pruebas unitarias de I05	Gian Carlos Huachín

Elaboración: El autor

Tabla 56. Sprint 4 de Identity

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de I06	Cristian Llanos
TDD	Programación de las pruebas unitarias de I07	Gian Carlos Huachín

Elaboración: El autor

Tabla 57. Sprint 5 de Identity

Sprint 5		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I01	Cristian Llanos
Implementación	Programación de la funcionalidad I02	Gian Carlos Huachín
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos

Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastian Burgos

Elaboración: El autor

Tabla 58. Sprint 6 de Identity

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I03	Cristian Llanos
Implementación	Programación de la funcionalidad I04	Gian Carlos Huachín
Implementación	Programación de la funcionalidad I05	Gian Carlos Huachín
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Cristian Llanos

Elaboración: El autor

Tabla 59. Sprint 7 de Identity

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad I07	Gian Carlos Huachín
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Cristian Llanos

Elaboración: El autor

Tabla 60. Sprint 8 de Identity

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a Producción	Sebastián Burgos
Pruebas	Pruebas en producción	Sebastián Burgos

Elaboración: El autor

Tabla 61. Sprint 9 de Identity

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea I08	Cristian Llanos
Refactorización	Desarrollo de la tarea I09	Giomar Rodriguez
Pruebas	Verificar que se pueda autenticar y registrar	Sebastián Burgos

Elaboración: El autor

Tabla 62. Sprint 10 de Identity

Sprint 10		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea I10	Giomar Rodriguez
Pruebas	Verificar que se pueda autenticar y registrar	Sebastián Burgos

Elaboración: El autor

Cronograma

El proyecto contó con 9 *sprints*, los cuales empezaron el 3 de septiembre del 2018 y finalizaron 30 de noviembre del 2018

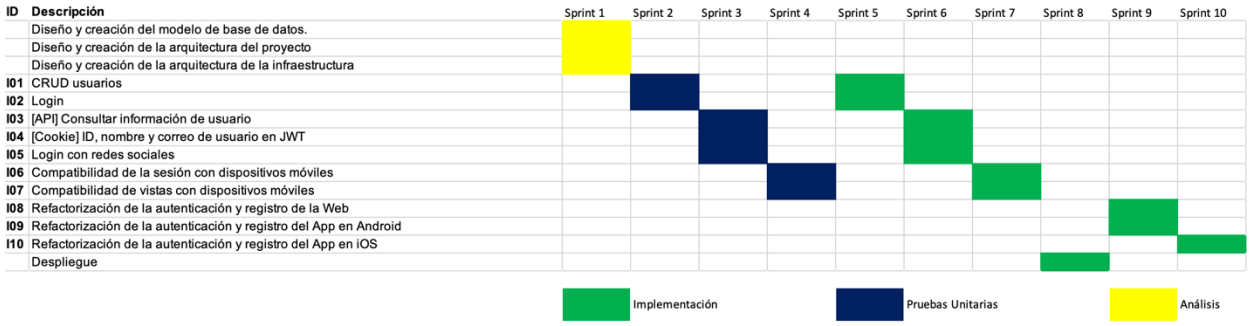

Figura 17. Cronograma Identity

Elaboración: El autor

Arquitectura

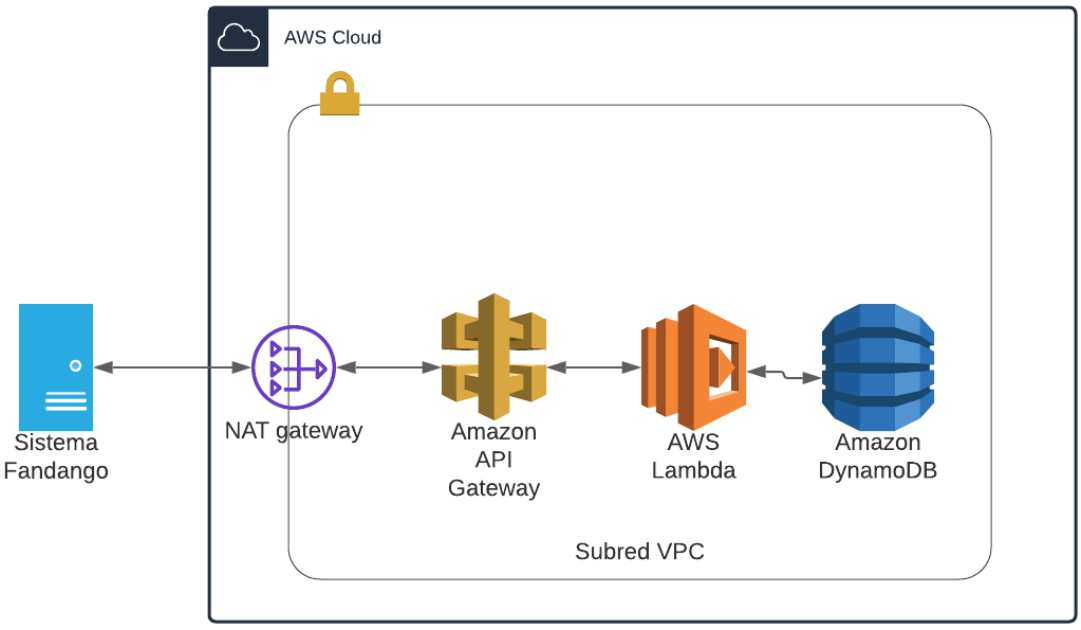

Figura 18. Arquitectura de Identity

Elaboración: El autor

Costos

Tabla 63. Costo mensual promedio de AWS de Identity

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
AWS Lambda	Cantidad de solicitudes (3000000)	USD$ 0.40
Amazon DynamoDB	Tamaño promedio del elemento (todos los atributos) (1 KB), Escribir el plazo de la capacidad reservada (1 año), Leer el plazo de la capacidad reservada (1 año), Tamaño del almacenamiento de datos (200 GB)	USD$ 162.98
Amazon API Gateway	Unidades de solicitudes de la API HTTP (millones), Tamaño promedio de cada solicitud (34 KB), Unidades de solicitud de la API REST (millones), Tamaño de memoria caché (GB) (Ninguno), Unidades de mensaje WebSocket (miles), Tamaño promedio del mensaje (32 KB), Solicitudes (3 por mes), Solicitudes (3 por mes)	USD$ 13.50
	Total	**USD$ 176.88**

Elaboración: El autor

5. POS

Necesidades

- La alta latencia y constantes caídas de los cines estaba generando problemas en la plataforma actual porque el sistema esperaba la respuesta del cine para generar una venta, ocasionando cuellos de botella en la conexión a la base de datos.

- Se necesita aislar el servicio, porque al ser el más importante, es al que se le da más recursos computacionales porque el modelo de negocio depende completamente de la venta de entradas al cine.

Stakeholders

Tabla 64. Stakeholders de POS

Entidad/Persona	**Descripción**
Fandango	Sistema que consumirá este servicio para la venta de entradas.
Exhibition Relations	Área de Fandango que trabaja la relación con los cines
Operaciones	Área de Fandango que ve supervisa las transacciones y la operatividad de los cines, tiene constante comunicación con los cines para mantener las ventas disponibles en todas las funciones.
Vista	POS de cine
Cinestar Software	POS de cine
E-Score	POS de cine
Ewave	POS de cine
UpBase	POS de cine
Sypsoft	POS de cine
Cinext	POS de cine

Cinebox	POS de cine
Cinemagix	POS de cine
Cinetix	POS de cine
Data	El equipo de Data es el que sincroniza la información de los cines con el sistema de Fandango.
Infraestructura	Gestión del servicio

Elaboración: El autor

Requisitos

- El lanzamiento se hará progresivo y por grupo de cines cada semana.
- En caso un cine esté presentando problemas se debe cerrar la conexión automáticamente y poder verificar cuando el servicio esté disponible nuevamente para reestablecer la conexión, utilizando Circuit Breaker Pattern (Netflix, 2017).
- El servicio debe abstraer la lógica de cada cine a una única interfaz, para que el que necesite generar una compra de asientos lo pueda hacer sin conocer cómo funciona cada cine.
- Mantener las funciones actualizadas todo el tiempo.
- Evitar doble venta del mismo asiento para la misma función.

Matriz de Responsabilidades

Tabla 65. Matriz de Responsabilidades de POS

Responsable	**Rol en el proyecto**	**Cargo**
Sebastián burgos	Líder del proyecto	CTO
Diego Michelena	Desarrollador	Líder del equipo de integraciones
Franz Cajahuanca	Desarrollador	Desarrollador
Andrés Sandoval	Desarrollador	Desarrollador
Melanie Velásquez	Data/Sincronización	Líder del equipo de data

Melanie Diaz	Tester	Analista del equipo de operaciones

Elaboración: El autor

Product Backlog

Tabla 66. Product Backlog de POS

ID	Descripción	Story Points
P01	Abstraer la interfaz de todos los cines	13
P02	Circuit Breaker	13
P03	[Vista] Obtener Precios	2
P04	[Vista] Obtener Mapa	2
P05	[Vista] Reservar Asientos	3
P06	[Vista] Comprar Ticket	3
P07	[Vista] Anular Ticket	3
P08	[Cinestar Software] Obtener Precios	2
P09	[Cinestar Software] Obtener Mapa	2
P10	[Cinestar Software] Reservar Asientos	3
P11	[Cinestar Software] Comprar Ticket	3
P12	[Cinestar Software] Anular Ticket	3
P13	[E-Score] Obtener Precios	2
P14	[E-Score] Obtener Mapa	2
P15	[E-Score] Reservar Asientos	3
P16	[E-Score] Comprar Ticket	3
P17	[E-Score] Anular Ticket	3
P18	[EWave] Obtener Precios	2
P19	[EWave] Obtener Mapa	2
P20	[EWave] Reservar Asientos	3
P21	[EWave] Comprar Ticket	3
P22	[EWave] Anular Ticket	3
P23	[UpBase] Obtener Precios	2
P24	[UpBase] Obtener Mapa	2

P25	[UpBase] Reservar Asientos	3
P26	[UpBase] Comprar Ticket	3
P27	[UpBase] Anular Ticket	3
P28	[SypSoft] Obtener Precios	2
P29	[SypSoft] Obtener Mapa	2
P30	[SypSoft] Reservar Asientos	3
P31	[SypSoft] Comprar Ticket	3
P32	[SypSoft] Anular Ticket	3
P33	[Cinext] Obtener Precios	2
P34	[Cinext] Obtener Mapa	2
P35	[Cinext] Reservar Asientos	3
P36	[Cinext] Comprar Ticket	3
P37	[Cinext] Anular Ticket	3
P38	[Cinebox] Obtener Precios	2
P39	[Cinebox] Obtener Mapa	2
P40	[Cinebox] Reservar Asientos	3
P41	[Cinebox] Comprar Ticket	3
P42	[Cinebox] Anular Ticket	3
P43	[Cinemagic] Obtener Precios	2
P44	[Cinemagic] Obtener Mapa	2
P45	[Cinemagic] Reservar Asientos	3
P46	[Cinemagic] Comprar Ticket	3
P47	[Cinemagic] Anular Ticket	3
P48	[Cinetix] Obtener Precios	2
P49	[Cinetix] Obtener Mapa	2
P50	[Cinetix] Reservar Asientos	3
P51	[Cinetix] Comprar Ticket	3
P52	[Cinetix] Anular Ticket	3
P53	Módulo de sincronización de películas	13
P54	Módulo de sincronización de funciones	13

P55	[API] precios	8
P56	[API] mapa	8
P57	[API] reserva	8
P58	[API] compra	8
P59	[API] anulación	8
P60	Refactorización de Fandango para usar el nuevo servicio	13

Elaboración: El autor

Sprint Backlog

Tabla 67. Sprint 1 de POS

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño y creación del modelo de base de datos.	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura del proyecto	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Sebastián Burgos

Elaboración: El autor

Tabla 68. Sprint 2 de POS

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño de las tramas de los servicios	Sebastián Burgos
TDD	Programación de las pruebas unitarias de P01	Diego Michelena
TDD	Programación de las pruebas unitarias de P02	Franz Cajahuanca

Elaboración: El autor

Tabla 69. Sprint 3 de POS

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de P03, P04, P05, P06, P07	Diego Michelena
TDD	Programación de las pruebas unitarias de P08, P09, P10, P11, P12	Franz Cajahuancha
TDD	Programación de las pruebas unitarias de P13, P14, P15, P16, P17	Franz Cajahuanca
TDD	Programación de las pruebas unitarias de P18, P19, P20, P21, P22	Diego Michelena
TDD	Programación de las pruebas unitarias de P23, P24, P25, P26, P27	Andrés Sandoval
TDD	Programación de las pruebas unitarias de P28, P29, P30, P31, P32	Andrés Sandoval

Elaboración: El autor

Tabla 70. Sprint 4 de POS

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de P33, P34, P35, P36, P37	Franz Cajahuanca
TDD	Programación de las pruebas unitarias de P38, P39, P40, P41, P42	Franz Cajahuanca
TDD	Programación de las pruebas unitarias de P43, P44, P45, P46, P47	Diego Michelena
TDD	Programación de las pruebas unitarias de P48, P49, P50, P51, P52	Diego Michelena
TDD	Programación de las pruebas unitarias de P53	Sebastián Burgos

TDD	Programación de las pruebas unitarias de P54	Sebastián Burgos
TDD	Programación de las pruebas unitarias de P55	Andrés Sandoval
TDD	Programación de las pruebas unitarias de P56	Andrés Sandoval

Elaboración: El autor

Tabla 71. Sprint 5 de POS

Sprint 5		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de P57	Diego Michelena
TDD	Programación de las pruebas unitarias de P58	Franz Cajahuanca
TDD	Programación de las pruebas unitarias de P59	Andrés Sandoval

Elaboración: El autor

Tabla 72. Sprint 6 de POS

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad P01	Sebastián Burgos
Implementación	Programación de la funcionalidad P02	Franz Cajahuanca
Implementación	Programación de la funcionalidad P03, P04, P05, P06, P07	Diego Michelena
Implementación	Programación de la funcionalidad P08, P09, P10, P11, P12	Andrés Sandoval
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos

Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Melanie Velasquez y Melanie Diaz
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 73. Sprint 7 de POS

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad P13, P14, P15, P16, P17	Sebastián Burgos
Implementación	Programación de la funcionalidad P18, P19, P20, P21, P22	Diego Michelena
Implementación	Programación de la funcionalidad P23, P24, P25, P26, P27	Franz Cajahuanca
Implementación	Programación de la funcionalidad P28, P29, P30, P31, P32	Andrés Sandoval
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Melanie Velasquez y Melanie Diaz
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 74. Sprint 8 de POS

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad P33, P34, P35, P36, P37	Sebastián Burgos
Implementación	Programación de la funcionalidad P38, P39, P40, P41, P42	Franz Cajahuanca

Implementación	Programación de la funcionalidad P43, P44, P45, P46, P47	Andrés Sandoval
Implementación	Programación de la funcionalidad P48, P49, P50, P51, P52	Diego Michelena
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Melanie Velasquez y Melanie Diaz
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 75. Sprint 9 de POS

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad P53	Sebastián Burgos
Implementación	Programación de la funcionalidad P54	Diego Michelena
Implementación	Programación de la funcionalidad P55	Diego Michelena
Implementación	Programación de la funcionalidad P56	Franz Cajahuanca
Implementación	Programación de la funcionalidad P57	Franz Cajahuanca
Implementación	Programación de la funcionalidad P58	Andrés Sandoval
Implementación	Programación de la funcionalidad P59	Andrés Sandoval
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Melanie Velasquez y Melanie Diaz
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 76. Sprint 10 de POS

Sprint 10		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a Producción	Sebastián Burgos
Pruebas	Pruebas en producción	Sebastián Burgos

Elaboración: El autor

Tabla 77. Sprint 11 de POS

Sprint 11		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea P60	Diego Michelena
Pruebas	Verificar que se pueda comprar entradas	Melanie Velasquez y Melanie Diaz

Elaboración: El autor

Cronograma

El proyecto contó con 11 *sprints*, los cuales empezaron el 7 de enero del 2019 y finalizaron 22 de marzo del 2019

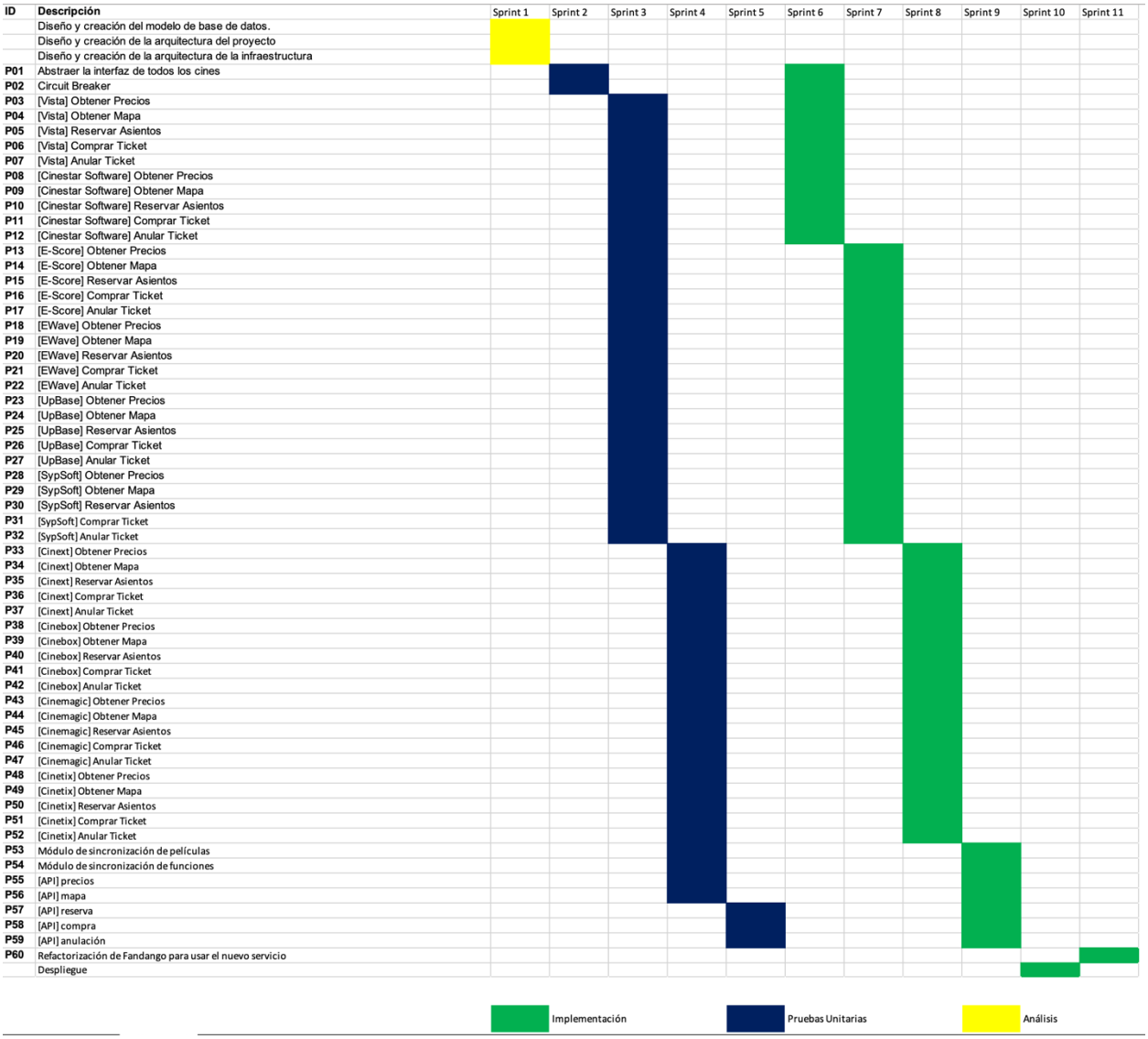

Figura 19. Cronograma POS

Elaboración: El autor

Arquitectura

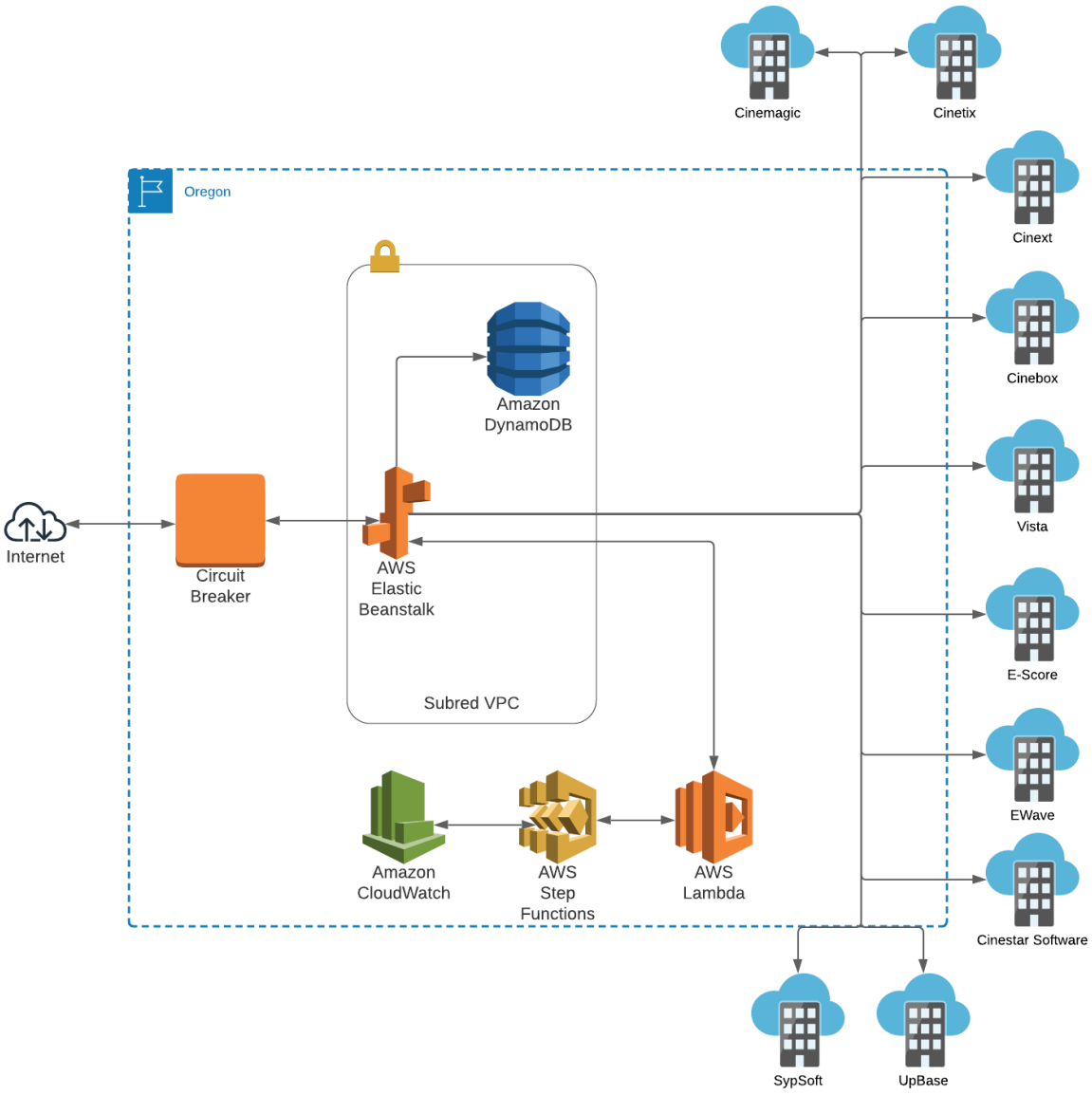

Figura 20. Arquitectura de POS

Elaboración: El autor

Costos

Tabla 78. Arquitectura de POS

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
Amazon CloudWatch	Número de eventos entre cuentas/personalizados (720)	USD$ 0.00
AWS Lambda	Cantidad de solicitudes (20000)	USD$ 0.00
AWS Step Functions	Solicitudes de flujo de trabajo (1 por hora), Transiciones de estado por flujo de trabajo (12)	USD$ 0.12
Amazon EC2	Sistema operativo (Linux), Cantidad (3), Pricing strategy (EC2 Instance Savings Plans 1 año Sin pagos iniciales), Cantidad de almacenamiento (30 GB), Tipo de instancia (t3.medium)	USD$ 66.16
Elastic Load Balancing	Número de balanceadores de carga de aplicaciones (1)	USD$ 600.43
Amazon DynamoDB	Tamaño promedio del elemento (todos los atributos) (1 KB), Escribir el plazo de la capacidad reservada (1 año), Leer el plazo de la capacidad reservada (1 año), Tamaño del almacenamiento de datos (200 GB)	USD$ 162.82
	Total	**USD$ 829.53**

Elaboración: El autor

6. Voucher

Necesidades

La complejidad de este servicio era alta por la volatilidad en la variación de cada tipo de promoción que se buscaba implementar, por esta razón este microservicio se dividió en etapas para poder aprender de esta variación en conjunto que se iba desarrollando. La primera etapa estuvo enfocada en aprender como se iban a ir dando estas reglas, fue una etapa muy manual pero que nos permitía iterar rápidamente para probar ideas que solucionen esa promoción y la segunda etapa estuvo centrada en automatizar las reglas y que estas sean lo suficientemente flexibles como para abarcar cualquier promoción, pero de igual forma el servicio era capaz de extenderse en funcionalidades con mucha sencillez.

Stakeholders

Tabla 79. Stakeholders de Voucher

Entidad/Persona	**Descripción**
Business Development	Área de Fandango que explora nuevos negocios.
Fandango	Sistema de Fandango que utilizará este sistema para verificar la promoción a otorgar
Infraestructura	Gestión del servicio en la nube
Coca Cola	Cliente interesado en usar nuestros códigos promocionales
Movistar	Cliente interesado en usar nuestros códigos promocionales
Wong	Cliente interesado en usar nuestros códigos promocionales

Elaboración: El autor

Requisitos

- Se debe poder generar múltiples códigos para una promoción.

- Se debe poder generar un solo código promocional para una campaña. Ejemplo: CocaCola20%
- Dado que las reglas son inciertas al inicio del proyecto, se sugiere partirlo en dos etapas para aprender como funcionarán las reglas antes de automatizar todo el proceso.
- Aunque en principio dado un código obtienes un descuento, en la práctica hay distintas reglas que dicen si es válido, como:
  - Día de la semana
  - Feriados
  - Tipo de Película
  - Estreno
  - Monto por descontar
  - Usó un descuento antes
  - Se puede acumular descuentos
  - Cines
  - Ciudad
  - País
- Se debe integrar este nuevo microservicio al sistema actual.

Matriz de Responsabilidades

Tabla 80. Matriz de Responsabilidades de Voucher

Responsable	**Rol en el proyecto**	**Cargo**
Sebastián Burgos	Líder del proyecto	CTO
Cristian Llanos	Desarrollador	Software Architect
Diego Humpire	Desarrollador	Desarrollador

Elaboración: El autor

Product Backlog

Tabla 81. Product Backlog de Voucher

ID	Descripción	Story Points
V01	CRUD Códigos	13
V02	CRUD Código Global	13
V03	Asignar reglas a grupo de códigos	8
V04	Regla: Día de la semana	5
V05	Regla: Feriado	5
V06	Regla: Tipo de película	5
V07	Regla: Estrenos	5
V08	Regla: Cines	5
V09	Regla: País y ciudad	5
V10	Regla: Monto por descontar	5
V11	Regla: Promoción única vez	13
V12	Regla: Acumular promociones	13
V13	Reporte de uso de promociones	8
V14	Dashboard de estado de promociones	13
V15	Login	8
V16	Registro	8
V17	[API] Validación de código	13
V18	Refactorizar funcionalidad para canjear códigos promocionales	13

Elaboración: El autor

Sprint Backlog

Tabla 82. Sprint 1 de Voucher

Sprint 1		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño y creación del modelo de base de datos.	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura del proyecto	Sebastián Burgos
Análisis	Diseño y creación de la arquitectura de la infraestructura	Sebastián Burgos

Elaboración: El autor

Tabla 83. Sprint 2 de Voucher

Sprint 2		
Tipo de Tarea	**Descripción**	**Responsables**
Análisis	Diseño de las tramas de los servicios	Sebastián Burgos
TDD	Programación de las pruebas unitarias de V01	Cristian Llanos
TDD	Programación de las pruebas unitarias de V02	Diego Humpire

Elaboración: El autor

Tabla 84. Sprint 3 de Voucher

Sprint 3		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de V03	Cristian Llanos
TDD	Programación de las pruebas unitarias de V04	Cristian Llanos

TDD	Programación de las pruebas unitarias de V05	Cristian Llanos
TDD	Programación de las pruebas unitarias de V06	Cristian Llanos
TDD	Programación de las pruebas unitarias de V07	Diego Humpire
TDD	Programación de las pruebas unitarias de V08	Diego Humpire
TDD	Programación de las pruebas unitarias de V09	Diego Humpire
TDD	Programación de las pruebas unitarias de V10	Diego Humpire

Elaboración: El autor

Tabla 85. Sprint 4 de Voucher

Sprint 4		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de V11	Cristian Llanos
TDD	Programación de las pruebas unitarias de V12	Diego Humpire

Elaboración: El autor

Tabla 86. Sprint 5 de Voucher

Sprint 5		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de V13	Cristian Llanos
TDD	Programación de las pruebas unitarias de V14	Diego Humpire

TDD	Programación de las pruebas unitarias de V15	Diego Humpire

Elaboración: El autor

Tabla 87. Sprint 6 de Voucher

Sprint 6		
Tipo de Tarea	**Descripción**	**Responsables**
TDD	Programación de las pruebas unitarias de V16	Cristian Llanos
TDD	Programación de las pruebas unitarias de V17	Diego Humpire

Elaboración: El autor

Tabla 88. Sprint 7 de Voucher

Sprint 7		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V01	Cristian Llanos
Implementación	Programación de la funcionalidad V02	Diego Humpire
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 89. Sprint 8 de Voucher

Sprint 8		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V03	Diego Humpire
Implementación	Programación de la funcionalidad V04	Diego Humpire

Implementación	Programación de la funcionalidad V05	Diego Humpire
Implementación	Programación de la funcionalidad V06	Cristian Llanos
Implementación	Programación de la funcionalidad V07	Cristian Llanos
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 90. Sprint 9 de Voucher

Sprint 9		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V08	Cristian Llanos
Implementación	Programación de la funcionalidad V09	Cristian Llanos
Implementación	Programación de la funcionalidad V10	Diego Humpire
Implementación	Programación de la funcionalidad V11	Diego Humpire
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 91. Sprint 10 de Voucher

Sprint 10		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V12	Diego Humpire
Implementación	Programación de la funcionalidad V13	Cristian Llanos

Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 92. Sprint 11 de Voucher

Sprint 11		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V14	Diego Humpire
Implementación	Programación de la funcionalidad V15	Cristian Llanos
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos
Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 93. Sprint 12 de Voucher

Sprint 12		
Tipo de Tarea	**Descripción**	**Responsables**
Implementación	Programación de la funcionalidad V16	Cristian Llanos
Implementación	Programación de la funcionalidad V17	Diego Humpire
Pruebas	Verificación de que las pruebas unitarias funcionen	Sebastián Burgos
Pruebas	Validación de que las funcionalidades cumplen con su *definition of done*	Cristian Llanos

Despliegue	Pase al ambiente de pruebas en la nube	Sebastián Burgos

Elaboración: El autor

Tabla 94. Sprint 13 de Voucher

Sprint 13		
Tipo de Tarea	**Descripción**	**Responsables**
Despliegue	Pase a Producción	Cristian Llanos
Pruebas	Pruebas en producción	Cristian Llanos

Elaboración: El autor

Tabla 95. Sprint 14 de Voucher

Sprint 14		
Tipo de Tarea	**Descripción**	**Responsables**
Refactorización	Desarrollo de la tarea V18	Diego Humpire
Pruebas	Verificar que las promociones funcionen	Sebastián Burgos

Elaboración: El autor

Cronograma

El proyecto contó con 14 *sprints*, los cuales empezaron el 1 de abril del 2019 y finalizaron 5 de julio del 2019

Figura 21. Cronograma Voucher

Elaboración: El autor

Arquitectura

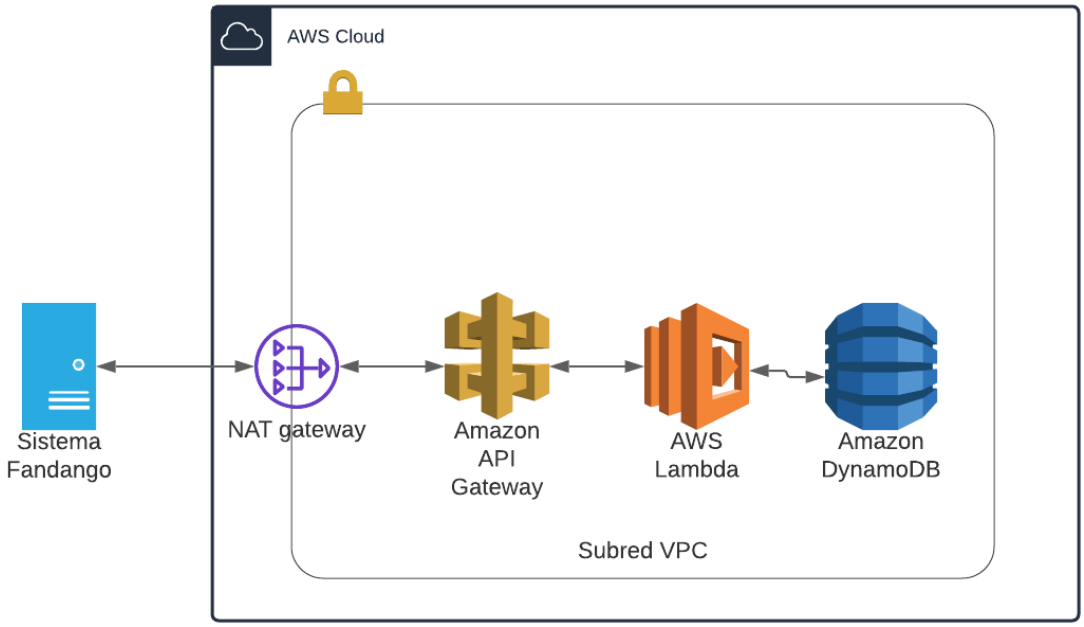

Figura 22. Arquitectura de Voucher

Elaboración: El autor

Costos

Tabla 96. Costo mensual promedio de AWS de Voucher

Costos Mensual Promedio de AWS		
Servicio	**Descripción**	**Costo Mensual**
Amazon DynamoDB	Tamaño promedio del elemento (todos los atributos) (1 KB), Escribir el plazo de la capacidad reservada (1 año), Leer el plazo de la capacidad reservada (1 año), Tamaño del almacenamiento de datos (100 GB)	USD$ 82.95
AWS Lambda	Cantidad de solicitudes (200000)	USD$ 0.00
AWS API Gateway	Unidades de solicitudes de la API HTTP (millones), Tamaño promedio de cada solicitud (34 KB), Unidades de solicitud de la API REST (millones), Tamaño de memoria caché (GB) (Ninguno), Unidades de mensaje WebSocket (miles), Tamaño promedio del mensaje (32 KB), Solicitudes (0.2 por mes)	USD$ 0.70
	Total	**USD$ 83.65**

Elaboración: El autor

Es importante recordar que cada uno de los servicios mencionados fueron desplegándose en medida que se iban terminando, esto nos dio una gran ventaja de ir solucionando problemas en el camino que permitieron mejorar la disponibilidad de la plataforma a un 100%, sobre todo por el problema de latencias y caídas de los cines.

Tabla 97. Costo de AWS por servicio (mensual)

Costos de AWS por servicio (mensual)	
Servicio	**Costo**
Monolito sin desacoplamiento	USD$ 17 080.79
Base de datos r3.8xlarge	(USD$ 11 605.92)
Base de datos r3.2xlarge	USD$ 3 230.44
Invoice	USD$ 402.16
Notification	USD$ 500.11
Blue	USD$ 225.60
Identity	USD$ 176.88
POS	USD$ 829.53
Voucher	USD$ 83.65
Total	**USD$ 10 923.24**

Elaboración: El autor

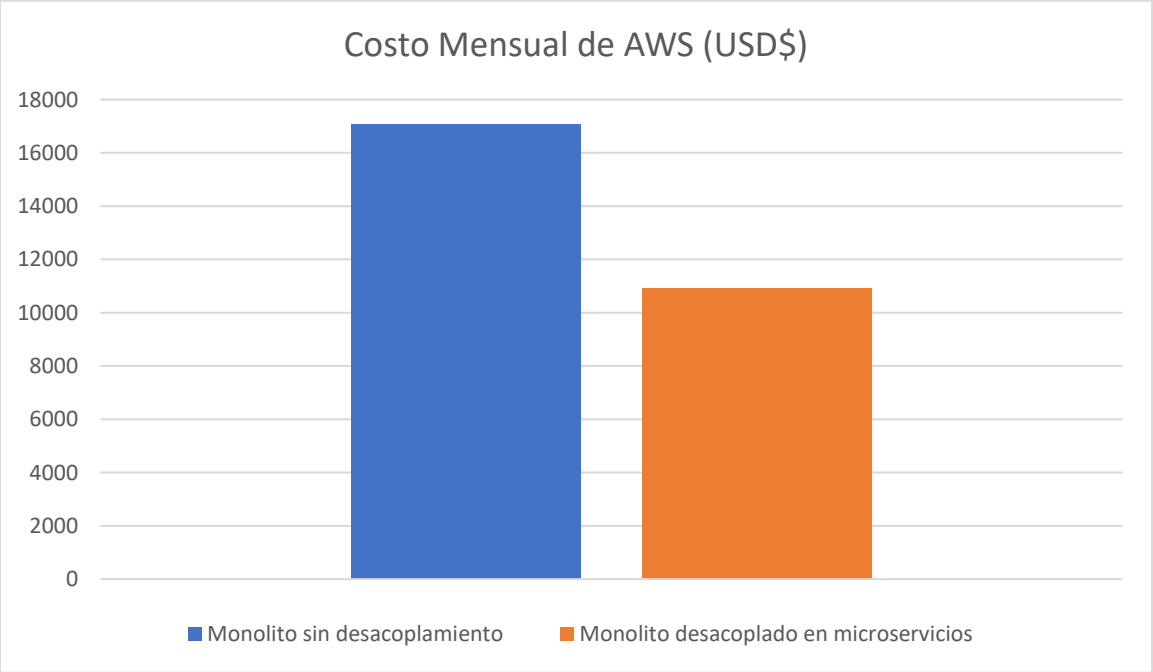

Figura 23. Costo Mensual de AWS (USD$)

Elaboración: El autor

Se pudo reducir la capacidad de las instancias de la base de datos a una del tipo r3.2xlarge, la cuál tiene un costo mensual de USD$ 3 230.44. Gracias a esta reducción y adicionando el costo de los nuevos servicios, se pudo reducir el gasto en un 36.05%, logrando sobrepasar el objetivo de disminuir en un 25% los costos en infraestructura.

3.4.6 Políticas de escalamiento

Necesidades

Dado que es completamente predecible las películas que están por estrenarse y el éxito que tendrán, también llamadas taquillazos o blockbusters. Asimismo, se sabía con anticipación el lanzamiento de campañas publicitarias como también de promociones con algún tercero, decidimos utilizar el escalamiento predictivo como estrategia. Del mismo modo, el hecho de tener la aplicación partida en pedazos más pequeños, microservicios, hizo que tengamos data más precisa sobre el uso de recursos en cada uno de los servicios.

Matriz de Responsabilidades

Tabla 98. Matriz de responsabilidades de Políticas de Escalamiento

Responsable	**Rol en el proyecto**	**Cargo**
Sebastián Burgos	Líder del proyecto	CTO
Jhon Campos	Infraestructura	Líder del equipo de infraestructura

Elaboración: El autor

Resultado

Luego de hacer un análisis del histórico y basándonos en variables como si hubiera una película de estreno que sería un taquillazo, la cuál traería a millones de personas a comprar la preventa, y si el área de marketing invertía en publicidad en esa misma semana. Esto nos permitió diseñar dos matrices las cuales facilitaban la toma de decisión para escalar en cada caso y poder prever el tiempo que esa tarea podría tomar en realizarse, usualmente esos escalamientos los hacíamos un día hábil antes del evento. También nos permitía delegar esta tarea a otras personas del equipo de infraestructura.

Tabla 99. Escalamiento en semana de estreno con taquillazo

Escalamiento en semana de estreno con taquillazo			
Servicio	**Con publicidad pagada**	**Sin publicidad**	**Tiempo**
Monolito	RDS de r3.2xlarge a r3.4xlarge Cantidad de EC2 de 4 a 12	Cantidad de EC2 de 4 a 8	4h
Invoice	RDS de r3.large a r3.xlarge Cantidad de EC2 de 2 a 6	Cantidad de EC2 de 2 a 4	3h
Blue	Escala solo por utilizar serverless*		
Identity	Escala solo por utilizar serverless*		
POS	Cantidad de EC2 de 4 a 12	Cantidad de EC2 de 4 a 8	30m
Voucher	Escala solo por utilizar serverless*		
Notification	Escala solo por utilizar serverless*		

Elaboración: El autor

Tabla 100. Escalamiento en semana de estreno sin taquillazo

Escalamiento en semana de estreno sin taquillazo			
Servicio	**Con publicidad pagada**	**Sin publicidad**	**Tiempo**
Monolito	Cantidad de EC2 de 4 a 6	Sin modificaciones	30m
Invoice	Cantidad de EC2 de 2 a 2	Sin modificaciones	30 m
Blue	Escala solo por utilizar serverless*		
Identity	Escala solo por utilizar serverless*		
POS	Cantidad de EC2 de 4 a 6	Sin modificciones	30m
Voucher	Escala solo por utilizar serverless*		
Notification	Escala solo por utilizar serverless*		

Elaboración: El autor

* Al utilizar tecnologías serverless, en este caso AWS Lambda con AWS API Gateway, la escalabilidad es determinada por el mismo Amazon y no debemos preocuparnos de esto. Va de cero cuando nadie utiliza el servicio

hasta las máximas posibles dentro de toda la infraestructura de Amazon Web Services (Amazon, 2021).

CAPÍTULO IV

REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

El aporte que se tuvo con el desacoplamiento del monolito en microservicios estuvo en adelantarnos a problemas graves en el futuro, identificando responsablemente los puntos más críticos del sistema, teniendo claro cuáles eran las prioridades del negocio y desde ahí planificar la desestructuración del sistema microservicios. En el pasado fue más sencillo crear un nuevo sistema y migrar toda la información, pero en medida de que el negocio se volvía más grande, los requerimientos generaban que no era posible hacer una pausa y correr el riesgo de la noche a la mañana migrar todo y esperar que funcione a la perfección nos hizo tomar la decisión de hacer una migración evolutiva y descentralizada, de esta forma cada servicio podía evolucionar independientemente en el tiempo hacia donde sus propias necesidades le exijan, tanto en funcionalidades como en escalamiento de servicios para tener una alta disponibilidad. También el aporte estuvo centrado en acompañar las decisiones de negocio exponiendo las necesidades como área de tecnología y profundizando que Fandango es una empresa de servicios tecnológicos que vende entradas al cine, tener una alta

disponibilidad era clave para satisfacer no solo a los consumidores finales sino también a los cines.

La realidad siempre fue que teníamos recursos limitados para hacer una gran migración, pero también éramos conscientes de que se tenía que lograr porque en medida que íbamos creciendo las probabilidades de caídas iban aumentando, lo más responsable fue dividir el problema y atacarlo primero en puntos sencillos para aprender y luego ir por esos módulos que eran bombas de tiempo para la estabilidad del sistema.

Entre las prácticas que ejecutamos estuvo primero entender el contexto en el que se encontraba la compañía donde se buscaba crecer aceleradamente, lo cuál hace que siempre tengas nuevas cosas que hacer, aunque suena bien, no te dejaba espacio para solucionar problemas del pasado. Luego, estuvo la planificación, dado el contexto en el que estábamos, debíamos tener claro en una hoja de ruta que cosas íbamos a hacer a lo largo del año, desde ahí priorizar todas las necesidades del negocio y agregar las necesidades como área de tecnología, aprendí mucho sobre negociación y persuasión para lograr que el negocio entienda la importancia de estos cambios. Como líder del área de tecnología, mi responsabilidad era que todo funcione como se esperaba. Por último, la ejecución era clave en este proceso, teníamos que ser exitosos en lo planificado, lo cuál significa cumplir con los alcances que nosotros mismos nos planteamos. En muchas partes del desacoplamiento me tocó volver a programar y ayudar a desatorar problemas de los cuales me eran muy familiares porque algunos venían de problemas con integraciones, área que lidere por muchos años.

Todo esto me permitió crecer profesionalmente en área donde era débil, las cuales eran la planificación y negociación. Como también en saber motivar al equipo en perseguir objetivos, lograrlos y dar los siguientes pasos. Fue muy gratificante lanzar a producción servicios que eran muy críticos para el sistema y luego poder ver reflejado esos cambios en costos más bajos en Amazon Web Service o llegar a picos de demanda muy altos y que no generaran ningún efecto negativo en nuestra infraestructura.

Estos logros me permitieron colocarme como uno de los “CTO – Founder” promesas del Perú (Mitchell, 2020), como también en el pasado poder haber dado la conferencia principal para el evento más importante de la Facultad de Ingeniería y Arquitectura de la Universidad de San Martín de Porres, Visión 2016 (USMP, 2016). Por último, mi experiencia desde Cinepapaya me permitió dar múltiples charlas y entrevistas a nivel nacional e internacional, ganando mucho prestigio a nivel de desarrolladores y emprendedores.

Tomar el liderazgo de un equipo de 28 personas fue un reto muy gratificante, pero que tomé con mucha responsabilidad, los tiempos fueron cortos y la velocidad en la que se movía el negocio me invitaron a ser muy atento con cada uno de los pasos que iba dando, esto hizo que aprovechara mucho mis espacios libres para auto capacitarme, leí mucho sobre liderazgo, planificación, gerencia, administración de negocios, planes estratégicos, gestión de equipo, gestión de conocimiento y coaching. Específicamente sobre el proyecto en mención, conversamos con nuestros pares en Fandango US e Ingresso en Brasil para absorber sus experiencias en procesos similares, leí mucho sobre otras empresas que desacoplaron sus monolitos en microservicios: patrones, buenas prácticas, metodologías, errores pasados, entre otros. Tuve un gran equipo, completamente capaz de asumir este reto con el profesionalismo necesario, estoy muy agradecido por haberlos tenido conmigo y lograr con éxito cada pase a producción de cada microservicio.

CONCLUSIONES

1. Se lograron migrar los servicios que más problemas daban al monolito, lo cuál permitió darles los recursos computacionales necesarios dependiendo del tráfico que tenía la plataforma, reduciendo y haciendo predecible la inversión que se le daba a cada servicio.

2. Al poder manejar independientemente los recursos de infraestructura que se le daba a cada servicio se pudo reducir los costos en 36.05%. Principalmente porque descongestionamos la base de datos principal del monolito a microservicios que utilizaban la tecnología adecuada para cada problema que intentaban resolver.

3. Debido a que pudimos aislar el servicio de venta de entradas (POS), y este ya no se afectaba por las caídas de los cines gracias al *Circuit Breaker Pattern*, desde su lanzamiento tuvimos el servicio disponible en un 100%, no sufrió ninguna caída. Con esto logramos satisfacer la necesidad del negocio de tener las ventas activas el mayor tiempo posible.

4. Aunque no pudimos finalizar con el desacoplamiento de todos los servicios debido al cierre de operaciones de Fandango Latinoamérica en el 2020, pudimos extraer del monolito los más problemáticos, los cuales fueron: Invoice, Blue, POS, Identity y Voucher.

5. El uso de la estrategia predictiva para el auto-escalamiento de la infraestructura nos ayudó a utilizar los recursos necesarios en el momento adecuado

dependiendo de si en la semana (de jueves a miércoles) había un estreno de un taquillazo y si Fandango invertía en publicidad, ambas variables se traducían en aumentos significativos de tráfico.

6. Cada lanzamiento a producción fue exitoso y dentro de las fechas establecidas, esto era clave para la continuidad de los siguientes proyectos, porque el negocio constantemente requería del equipo de tecnología para agregar valor al producto actual.

RECOMENDACIONES

1. Desde el inicio de un proyecto debemos intentar construirlo de tal manera que sea fácilmente desacoplable, no recomendaría partir de una estructura de microservicios por la complejidad en el mantenimiento de estos, y porque una de las cosas más importantes en una Startup es salir a producción lo más pronto posible e iterar hasta encontrar el “Product Market Fit”.

2. El desacoplamiento debe ser progresivo, como se demostró en el informe, esta decisión permitió aprender rápidamente y con un margen de error pequeño en cada despliegue porque era muy sencillo hacer Rollback.

3. Debemos pensar que estos procesos pueden ser nuevos para muchos en el equipo, por lo que es importante seleccionar correctamente los servicios con los que empezaremos para que el aprendizaje sea suficientemente bueno, pero con un riesgo bajo ante errores.

4. Demostrarle al negocio los avances que se van logrando es una buena forma de mantener la priorización de los siguientes microservicios que se van a desarrollar, es importante involucrar a las otras áreas y que este proyecto se vea como una evolución.

5. El desacoplamiento de servicios o cualquier refactorización debe hacerse porque realmente estamos anticipándonos a un problema en un futuro cercano, el cuál

responsablemente debemos resolver a tiempo, y no debe hacerse por una motivación de rehacer todo por emplear nuevas tecnologías.

FUENTES DE INFORMACIÓN

Bibliográficas:

Martin, R. C. (2008). Clean Code: A Handbook of Afile Software Craftsmanship. Bostón. Pearson; 1era edición.

Newman, S. (2015). Building Microservices (Vol. Primera edición). California. O'Reilly.

Hemerográficas:

Singh, V., & Peddoju, K. S. (2017). Container-based microservice architecture for cloud applications. 2017 International Conference on Computing, Communication and Automation (ICCCA), 847-852.

Al-Debagy, O., & Martinek, P. (2018). A Comparative Review of Microservices and Monolithic Architectures. 18th IEEE International Symposium on Computational Intelligence and Informatics. IEEE, Budapest, Hungary.

Macedo, A. (Junio de 2020). Uso de una arquitectura basada en eventos como capa de comunicación para microservicios. Obtenido de http://tesis.pucp.edu.pe/repositorio/bitstream/handle/20.500.12404/16979/MACEDO_PEREIRA_ALEJANDRO_USO_UNA_ARQUITECTURA.pdf?sequence=1&isAllowed=y

Arboleda Cola, C. A. (Noviembre de 2017). Propuesta metodológica para migración de sistemas Web con arquitectura monolítica hacia una arquitectura basada en micrsoservicios. Obtenido de https://bibdigital.epn.edu.ec/handle/15000/18989

God, K., & Zabierowski, W. (2020). The Comparison of Microservice and Monolithic Architecture. 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). IEEE.

Ochoa, L., Villamizar, M., Castro, H., Garces, O., Salamanca, L., Verano, M., . . . Lang, M. (2016). Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures. 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid). Bogotá: IEEE.

Ynga, C., & Palacios, P. (2015). PROPUESTA DE IMPLEMENTACIÓN de un marco de trabajo para el desarrollo de aplicaciones ANDROID. Obtenido de http://hdl.handle.net/10757/556495

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation. California: University of California, Irvine.

Garcia Fuentes, O. A. (2019). Análisis, Desarrollo e Implementación del Sistema Integral Universitario (SIU) para la USMP - Filial Norte. Lima.

Hema, V., Thota, S., Kumar, S., Padmaja, C., Krishna, C., & Mahender, K. (2020). Scrum: An Effective Software Development Agile Tool. International Conference on Recent Advancements in Engineering and Management (ICRAEM-2020) 9-10 October 2020, Warangal, India. IOP Conference Series: Materials Science and Engineering.

Siniaalto, M., & Abrahamsson, P. (2007). A comparative case study on the impact of Test-Driven Development on Program Design and Test Coverage. IEEE.

Khanam, Z., & Ahsan, M. N. (2017). Evaluating the effectiveness of test driven development: Advantages and pitfalls. International Journal of Applied Enfineering Research.

Biswas, A., Mahumdar, S., Nandy, B., & El-Haraki, A. (2017). A hybrid auto-scaling technique for clouds processing applications with service level agreements. Journal of Cloud Computing, 29.

Electrónicas:

Wowza. (2014). Revisión general del producto Wowza Streaming Engine. Obtenido de Wowza: https://www.wowza.com/uploads/wowza4/documents/WowzaStreamingEngine-Product-Overview_final.pdf

The Org. (2021). The Org. Obtenido de Organigrama de NBC Universal: https://theorg.com/org/nbcuniversal/team

The Org. (2021). The Org. Obtenido de Organigrama Fandango LLC: https://theorg.com/org/fandango/team

Rest Api Tutorial. (2021). Rest Api Tutorial. Obtenido de https://www.restapitutorial.com/lessons/httpmethods.html

Dehgani, Z. (2018). Martin Fowler. Obtenido de https://martinfowler.com/articles/break-monolith-into-microservices.html

Mitchell, G. (2020). Ruta Startup. Obtenido de https://ruta-startup.com/preparing-for-a-new-wave-of-startup-founders-in-peru/

USMP. (2016). Visión 2016. Obtenido de https://www.usmp.edu.pe/vision2016/

Fowler, M., & Lewis, J. (25 de Marzo de 2014). Martin Fowler. Obtenido de https://martinfowler.com/articles/microservices.html

Dávila, L. (2019). Metodología de Desarrollo de Sistemas de Información. Obtenido de https://cdn.www.gob.pe/uploads/document/file/313911/Resoluci%C3%B3n_de_Secretar%C3%ADa_General_N__020-2019-PRODUCESG20190515-27906-1x5ijoj.pdf

Stoll, J. (21 de Abril de 2021). Number of Netflix paid subscribers worldwide from 1st quarter 2013 to 1st quarter 2021. Obtenido de https://www.statista.com/statistics/250934/quarterly-number-of-netflix-streaming-subscribers-worldwide/

Monolith First. (3 de Junio de 2015). Obtenido de Martin Fowler: https://martinfowler.com/bliki/MonolithFirst.html

Scrum Org. (s.f.). Obtenido de https://www.scrum.org/resources/what-is-scrum

eMarketer. (29 de Junio de 2017). El uso del smartphone en Latinoamérica. Obtenido de Statista: https://es.statista.com/grafico/10071/la-generalizacion-del-smartphone-en-latinoamerica/

Netflix. (3 de Julio de 2017). Github. Obtenido de https://github.com/Netflix/Hystrix/wiki/How-it-Works

Schwaber, K., & Sutherland, J. (2018). The Scrum Guide.

Scrum Inc. (2019). Obtenido de https://www.scruminc.com/definition-of-done/

Amazon Web Services. (2021). Predictive scaling for Amazon EC2 Auto Scaling. Obtenido de Amazon: https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-predictive-scaling.html

Gimenez, F. (08 de junio de 2020). Top five aws scaling strategies. Obtenido de Mission Cloud: https://www.missioncloud.com/blog/top-five-aws-auto-scaling-strategies

Amazon. (2021). Serverless. Obtenido de Amazon: https://aws.amazon.com/es/serverless