UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Transformación Digital en el sistema bancario: Un análisis Técnico-Económico de la aplicación de Backend For Frontend (BFF) en las arquitecturas de software.

Daniel Fernando Silva Espín

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación.

Quito, 15 de diciembre de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Transformación Digital en el sistema bancario: Un análisis Técnico-Económico de la aplicación de Backend For Frontend (BFF) en las arquitecturas de software.

Daniel Fernando Silva Espín

Nombre del profesor, Título académico **Ricardo Flores, PhD**

Quito, 15 de diciembre de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Daniel Fernando Silva Espín

Código: 00201725

Cédula de identidad: 1722865407

Lugar y fecha: Quito, 15 de diciembre de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En la actual era de transformación digital, el panorama empresarial experimenta cambios profundos impulsados por la adopción de nuevas tecnologías, que reconfiguran las operaciones y la interacción con los clientes. La industria bancaria, históricamente a la vanguardia de los avances tecnológicos, se esfuerza constantemente por mejorar los servicios y reforzar la seguridad para mantenerse competitiva en el ámbito financiero.

Es crucial que el diseño y la arquitectura del software sean óptimos, modulares y sostenibles en el tiempo, considerando las necesidades de mantenimiento y actualización. Se observa que algunos bancos han adoptado arquitecturas que aplican el concepto de Backend For Frontend (BFF), marcando un cambio respecto a enfoques más tradicionales. Dentro del marco de este proyecto, se implementará una metodología de trabajo estructurada que abarca diversas etapas, desde la revisión inicial del estado del arte hasta la implementación y evaluación de un prototipo de solución CRUD en un entorno de desarrollo con el uso de BFF. Los resultados de esta investigación se analizaron con el propósito de determinar la viabilidad y los beneficios de la implementación de BFF en el ámbito financiero. El objetivo general de este estudio es realizar un análisis técnico-económico sobre el uso de Backend For Frontend en la arquitectura de software para el desarrollo de soluciones digitales en servicios bancarios. Los objetivos específicos incluyen describir arquitecturas de desarrollo de software aplicables al sistema financiero, crear un prototipo de una solución de software CRUD utilizando BFF, establecer una base de datos para pruebas del prototipo, evaluar el rendimiento de la solución BFF y proporcionar un informe con las conclusiones del análisis. Este estudio busca aportar valiosos conocimientos en el ámbito de la arquitectura de software en constante evolución dentro del dinámico sector de servicios financieros.

ABSTRACT

In the current era of digital transformation, the business landscape is undergoing profound changes driven by the adoption of new technologies, reshaping operations and customer interactions. The banking industry, historically at the forefront of technological advances, continually strives to enhance services and reinforce security to remain competitive in the financial realm.

It is crucial that the design and architecture of software be optimal, modular, and sustainable over time, considering the needs for maintenance and updates. It is observed that some banks have adopted architectures applying the concept of Backend For Frontend (BFF), marking a shift from more traditional approaches. Within the framework of this project, a structured working methodology will be implemented, covering various stages from the initial review of the state of the art to the implementation and evaluation of a CRUD solution prototype in a development environment using BFF.

The results of this research were analyzed to determine the feasibility and benefits of implementing BFF in the financial domain. The overall objective of this study is to conduct a technical-economic analysis of the use of Backend For Frontend in software architecture for the development of digital solutions in banking services. Specific objectives include describing software development architectures applicable to the financial system, creating a prototype of a CRUD software solution using BFF, establishing a database for prototype testing, evaluating the performance of the BFF solution, and providing a report with the conclusions of the analysis. This study aims to contribute valuable insights into the constantly evolving field of software architecture within the dynamic sector of financial services.

Contenido

1. INTRODUCCION

En la actualidad, la transformación digital está cambiando al mundo. La industria ha empezado a implementar nuevas tecnologías que han revolucionado la forma en que operan los negocios y se relacionan con los clientes. La industria bancaria, a lo largo del tiempo, ha sido una de las que más se ha desarrollado tecnológicamente. Esto con el fin de proporcionar mejores servicios y mayor seguridad a sus clientes, con el objetivo de mantenerse competitivo en el área financiera. A lo largo del tiempo, se han visto una gran cantidad de servicios innovadores en el sector bancario. La alta competitividad en este campo ha llevado a que la transformación digital sea un factor determinante para la supervivencia y éxito de las instituciones financieras. La búsqueda de mejoras y la adopción de tecnologías avanzadas han redefinido por completo la forma en que los bancos operan y cómo los clientes interactúan con ellos. La banca móvil, servicios en línea, implementación de alta seguridad en todos sus productos y muchos más ejemplos de la forma en la que las entidades financieras se han adaptado al cambio tecnológico, además de que son pioneras en innovación.

El desarrollo de software es uno de los pilares principales en el crecimiento del sistema financiero. Esta afirmación se respalda en la capacidad del software para impulsar la innovación, mejorar la eficiencia operativa y enriquecer la experiencia del cliente en el sector bancario. En este contexto, es importante que el diseño y la arquitectura con la que se produce software sea óptimo, modular y que se pueda mantener con el tiempo, considerando el mantenimiento y actualización. Correspondiente del tiempo, las instituciones financieras han aplicado diferentes filosofías de diseño en sus soluciones de software. Sin embargo, en la actualidad se ha identificado que ciertos bancos están usando arquitecturas que aplican *Backend For Frontend* (BFF), dejando de lado otras que han

sido implementadas por mucho tiempo. Este hecho motiva a importante que se realice una investigación sobre este tema, e identificar los beneficios técnicos y económicos del uso de BFF en las instituciones financieras.

Objetivo General

Realizar un análisis técnico-económico sobre el uso de *Backend For Frontend* en la arquitectura de software para el desarrollo de soluciones digitales en servicios bancarios.

Objetivos Específicos

- Describir arquitecturas de desarrollo de software aplicables al sistema financiero.
- Realizar un prototipo de una solución de software CRUD usando BFF.
- Crear una base de datos para pruebas de prototipo.
- Realizar evaluación de desempeño de la solución BFF.
- Proporcionar un informe que incluye las conclusiones del análisis realizado.

Alcance

Este proyecto tiene como fin proporcionar una base sólida para la toma de decisiones estratégicas en la industria bancaria al evaluar el patrón de diseño *Backend For Frontend* (BFF) bajo la filosofía de la arquitectura de software de microservicios.

Metodología de trabajo

En el marco de este proyecto, se llevará a cabo una metodología de trabajo estructurada que abarca diversas etapas, desde la revisión inicial del estado del arte hasta la implementación y evaluación de un prototipo de solución CRUD en un entorno de desarrollo con el uso de *Backend For Frontend* (BFF).

2. ESTADO DEL ARTE

Las nuevas tecnologías desarrolladas a lo largo del tiempo como Big Data o Blockchain, han permitido la búsqueda de diferentes alternativas aplicadas a la transformación digital y la integración de tecnología, con la finalidad de impulsar a las entidades bancarias a la digitalización de procesos para mejorar la eficiencia en toda la empresa (Khanchel, 2019). La transformación digital de la industria bancaria actualmente puede tener una alta importancia para que los bancos se adapten a los cambios en los hábitos de los consumidores. Los nuevos participantes en el mercado ofrecen servicios financieros similares a los de la banca tradicional, pero no están sujetos a la misma regulación (Minin, 2018).

La evaluación de la calidad de la arquitectura de software para un sistema de banca en línea se basa en la importancia de la arquitectura de software para cumplir con los requisitos de calidad funcionales y no funcionales, y se presenta una comparación de diferentes métodos de evaluación de arquitectura. (Meiappane, 2013). En el caso de la banca en línea, los requisitos funcionales podrían incluir operaciones clave como la gestión de cuentas, transferencias de fondos, procesamiento de transacciones y generación de informes financieros.

Un panorama de aplicación de esta área es FinTech, una empresa financiera que explora los cambios y oportunidades que ha traído la digitalización en la industria bancaria, y cómo se adaptan al nuevo panorama competitivo (Omarini, 2017). Del mismo modo es importante la evaluación de la perspectiva gerencial sobre las barreras al cambio en la industria bancaria en relación con la transformación digital, para la búsqueda de las mejores decisiones para el diseño de la arquitectura aplicada (Diener, 2021).

2.1 APLICACIÓN DE BACKEND FOR FRONTEND (BFF) EN ARQUITECTURAS DE SOFTWARE BANCARIO: UN ENFOQUE TÉCNICO-ECONÓMICO

2.1.1 El negocio de la industria financiera

Como ya se mencionó, en el presente informe se pretende hacer un análisis del uso de BFF en la arquitectura de soluciones web de software en el contexto de instituciones financieras. Para esto, es importante entender a qué se dedican estas empresas, esto con el objetivo de posteriormente comprender por qué estas compañías han tenido un desarrollo tecnológico tan grande en los últimos 25 años.

Para empezar, se debe especificar que, la actividad principal de un banco consiste en intermediar entre quienes tienen dinero y quienes lo necesitan, obteniendo un beneficio por la diferencia entre los intereses que reciben y los que pagan. Los bancos pueden conseguir dinero de diversas formas, como los ahorros de los clientes, las cuentas a la vista, la emisión de bonos o el capital social.

Los bancos pueden prestar dinero a diferentes tipos de clientes, como individuos, empresas, gobiernos o entidades financieras. Los préstamos pueden tener diferentes características, como créditos al consumo, hipotecarios, empresariales o corporativos. Los bancos también pueden brindar otros servicios financieros, como el arrendamiento financiero, el descuento de facturas, las tarjetas de crédito o los seguros.

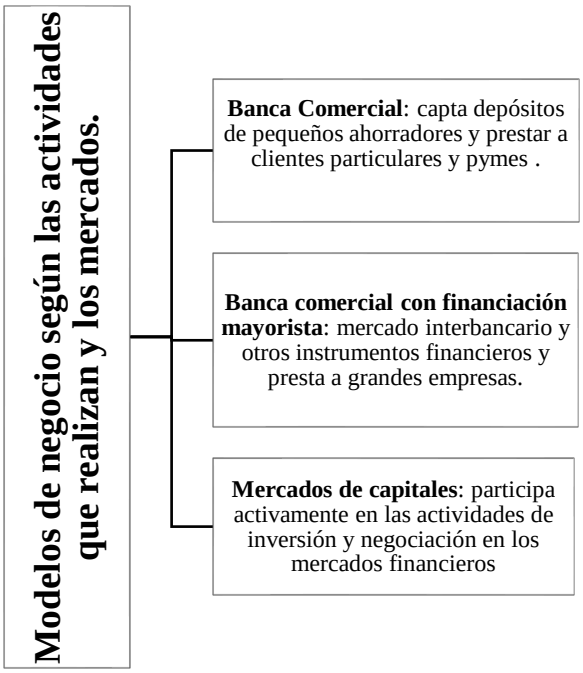

Ilustración 1 Modelos de negocio de la banca.

2.1.2 Transformación digital

Es necesario definir lo que es transformación digital. De forma breve esta puede ser definida como un proceso integral y estratégico que una organización emprende para aprovechar las oportunidades que brindan las tecnologías digitales en todos los aspectos de su operación, cultura y relaciones con sus clientes. Este proceso implica la reevaluación y la reestructuración de la infraestructura tecnológica existente, así como la redefinición de los modelos de negocio y procesos internos para maximizar la eficiencia, la innovación y el valor agregado.

En esencia, la transformación digital busca la integración y la optimización de las tecnologías digitales en todas las facetas de una empresa, con el objetivo de mejorar la toma de decisiones, aumentar la productividad, impulsar la agilidad organizativa y mejorar la experiencia del cliente. Esto puede incluir la adopción de herramientas de análisis de datos avanzados, la automatización de tareas, la implementación de la nube,

la creación de soluciones móviles, la incorporación de inteligencia artificial y la optimización de la ciberseguridad.

La transformación digital no solo implica tecnología, sino también aspectos culturales y organizativos. Requiere un cambio en la mentalidad de la empresa, promoviendo la colaboración, la innovación y la agilidad, y fomentando la adopción de una cultura orientada hacia datos. Esto implica la capacitación y el empoderamiento de los empleados para que utilicen las herramientas digitales de manera efectiva y participen activamente en la innovación. La transformación digital tiene como objetivo final posicionar a la organización en un lugar de ventaja competitiva en un entorno empresarial cada vez más digitalizado. Es un proceso continuo que se adapta constantemente a medida que evolucionan las tecnologías y las necesidades del mercado, y requiere un compromiso a largo plazo por parte de la organización para mantenerse relevante y competitiva en la era digital. “La transformación digital es la razón por la cual tantas áreas de negocio y de la vida son fundamentalmente diferentes a las de hace 20 años.” (IBM, 2023)

Aguirre nos indica que la transformación digital es: “La Transformación Digital puede ser definida como la adaptación continua a los cambios y usos de la tecnología; resultando en la modificación de los modelos de negocio que desarrollan las empresas, alineándose al avance en tecnología y a la dinámica de innovación propias a cada industria. A través del proceso, se evidencia una particular implicancia en los cambios en el comportamiento de los consumidores y como se relacionan con las empresas.” (Aguirre, 2021)

2.1.3 Transformación digital en el sector financiero.

La creciente penetración de internet y la difusión de los teléfonos móviles han modificado profundamente la forma en que los consumidores eligen interactuar con los servicios bancarios. En este sentido, los consumidores han adoptado con entusiasmo la práctica de

llevar a cabo sus transacciones financieras y compartir información personal a través de canales digitales. La disponibilidad de una conectividad constante y accesible ha disminuido las barreras geográficas y temporales, permitiendo a los consumidores acceder a servicios financieros digitales desde cualquier ubicación y en cualquier momento.

Los dispositivos móviles de última generación, especialmente los smartphones, han desempeñado un papel crucial en la modificación de estos hábitos. La accesibilidad asequible de redes móviles de alta velocidad, como 3G y 4G, ha habilitado a los consumidores para realizar operaciones bancarias en línea desde sus dispositivos móviles con una facilidad sin precedentes. Estos dispositivos han transformado los teléfonos en herramientas bancarias multifuncionales, permitiendo a los usuarios gestionar sus finanzas en cualquier momento y lugar. Asimismo, el informe resalta que los consumidores están buscando activamente nuevas formas de interactuar con los servicios financieros. Esta demanda ha impulsado a las entidades bancarias a desarrollar canales y productos digitales innovadores para atender estas necesidades cambiantes. La adaptación es fundamental para satisfacer las expectativas de los consumidores y mantener la relevancia en un mercado altamente competitivo. Las entidades bancarias que han avanzado significativamente en sus estrategias de digitalización están mejor preparadas para satisfacer las cambiantes demandas de sus clientes. Estas instituciones tienen la capacidad de responder de manera más ágil a las necesidades del mercado y ofrecer soluciones financieras innovadoras. La digitalización es esencial para mantener la competitividad en un entorno en el que emergen nuevos proveedores de servicios financieros digitales.

Para visualizar de mejor manera la tendencia ascendente del uso de medios electrónicos, se recopiló información sobre la cantidad de transacciones digitales realizadas en distintos países de Latinoamérica y que porcentaje representa en el total de transacciones. Con esto podemos ver que el uso de estas soluciones digitales es sumamente alto en varios de los países de Latinoamérica.

Tabla 1. Transacciones digitales en algunos países de la región

País	**Porcentaje de transacciones digitales**	**Número de transacciones digitales (millones)**
Argentina	64%	1.200
Brasil	75%	6.500
Chile	68%	1.100
Colombia	69%	3.400
México	62%	2.300
Perú	58%	700
Ecuador	41%	285

2.1.3.1 Empresas Fintech

Las empresas FinTech son altamente flexibles y pueden adaptarse rápidamente a los cambios en el entorno financiero. Además, suelen operar con estructuras de costos más bajas en comparación con los bancos tradicionales, lo que las hace altamente competitivas. Las plataformas de crowdfunding financiero y las criptomonedas tienen el potencial de eliminar por completo la necesidad de intermediación bancaria en ciertos casos, lo que puede tener un impacto altamente disruptivo en los paradigmas tradicionales. Según Cuesta, “la inversión en el sector alcanzó aproximadamente los

3.000 millones de dólares en el año 2013, lo que demuestra el creciente interés de los inversores en estas empresas. Esta financiación adicional ha permitido a las FinTech expandirse y desarrollar aún más sus innovadoras soluciones financieras". (Cuesta, 2015)

La interacción entre las FinTech y los bancos tradicionales promete moldear de manera significativa el futuro de la industria financiera y la forma en que se atienden las necesidades de los consumidores en la era digital (Forbes, 2020). "La digitalización ofrece nuevas oportunidades para que los bancos coloquen al cliente en el centro del proceso de desarrollo. Las nuevas tecnologías parecen estar y permanecer en el mercado para alterar la cadena de valor de los servicios financieros minoristas, además de introducir las Fintech en el ámbito competitivo. Las actuales y los recién llegados tienen palancas innovadoras que adoptar. Las fuerzas que configuran estos cambios han llevado al sector a asumir el papel de banca y finanzas, proveedor de servicios financieros." (Khanchel, 2019)

2.1.3.2 Transformación hacia la banca digital

Cuesta nos indica que la banca digital se puede definir como: "las formas de como generar la oferta, distribución y venta de productos y servicios financieros a través de canales digitales, explotar la última tecnología para conocer mejor a los clientes y anticiparse de forma rápida y adecuada a sus necesidades. Además, de crear la posibilidad de que los clientes se comuniquen con su banco a través de todos los canales, tanto analógicos como digitales, así como la automatización de los servicios." (Aguirre, 2021) En este contexto, los bancos convencionales que se embarcan en la banca digital están experimentando un cambio significativo.

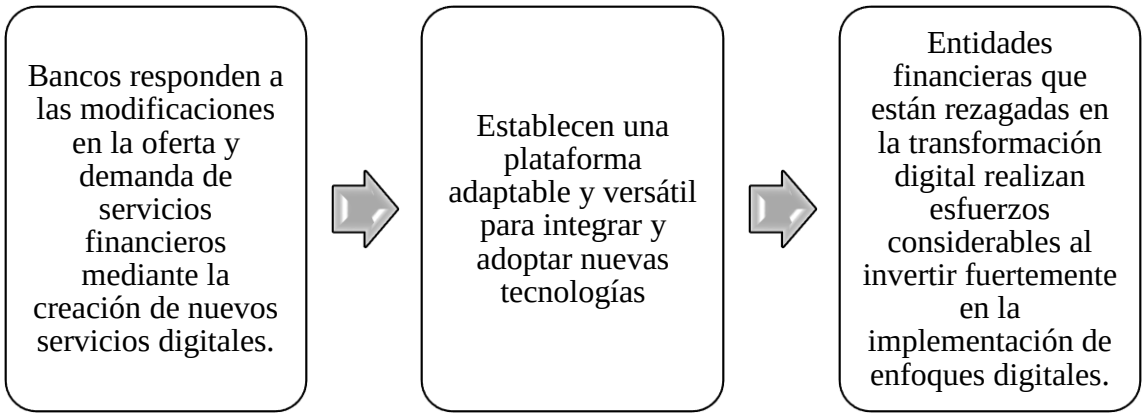

Ilustración 2 Etapas de evolución digital.

2.1.3.3 Reacción a la nueva competencia

Desde hace 25 años, instituciones financieras empezaron a proporcionar servicios en línea, lo que hecho que los clientes puedan acceder a una selección de sus productos financieros. En tiempos recientes, se ha hecho énfasis en la distribución mediante dispositivos móviles con el uso de apps.

Las soluciones en las que más han innovado los bancos son sobre los clientes finales. Los productos que más se han diversificado incluyen carteras digitales, soluciones de pago mediante tecnología de campo cercano (NFC) y aplicaciones para transferencias de dinero entre individuos (P2P).

2.1.3.4 Adaptación tecnológica

En el proceso de transformación tecnológica en el sector bancario se requiere: “llevar a cabo una renovación de la plataforma tecnológica, para convertirla en una infraestructura más modular y flexible que permita la integración de nuevas tecnologías, así como un desarrollo más rápido de nuevos productos.” (Cuesta, 2015) Es decir que, en este proceso de transformación digital, es fundamental un cambio a una organización tecnológica que se adapte con el tiempo.

Las soluciones digitales de las entidades financieras son exigidas a tener una gran capacidad de procesamiento de data y de peticiones. Además, es importante que las soluciones estén disponibles en varios canales y que sean intuitivas para el uso del cliente todos los días, a toda hora. Por lo que arquitectura debe ser lo suficientemente modular, con el fin de incorporar rápidamente las nuevas tecnologías o ajustarse a los cambiantes requisitos empresariales.

En este punto, las instituciones han pensado en que el desarrollo de tecnología debe ir también enfocado en procesos internos, en tareas que se hacen de forma manual y son monótonas, con la finalidad de mejorar la eficiencia. Además, el uso de tecnologías como la inteligencia artificial para la mejora de procesos de selección de productos y demás servicios a los clientes adecuados, y de esta forma poder entregar atención personalizada y por ende mejorar experiencia del cliente.

2.1.3.5 Efectos de la transformación digital en bancos

- Mejora en la experiencia del cliente, más rápida y eficiente. Lo que hace que el consumidor se fidelice con la marca.
- Menos contacto directo con el consumidor ya que al automatizar procesos el cliente no debe asistir de forma presencial para realizar cualquier tipo de trámite, a diferencia de la banca tradicional.
- Lo empleados cambian de actividades administrativas, a otras que tengan relación con el cliente para que generar mayor valor.
- Alta competitividad entre bancos, lo que hace que se presenten nuevas soluciones y servicios innovadores con el fin de captar nuevos consumidores.

2.1.4 Arquitectura de software

Bass nos indica que: “Una arquitectura de software es el diseño fundamental y la estructura subyacente que define cómo se construirá y operará un sistema de software. Es una representación abstracta que define los componentes del sistema, sus interacciones y las reglas que gobiernan su funcionamiento. En esencia, la arquitectura de software es el plano maestro que guía el desarrollo de la aplicación, proporcionando una visión clara de cómo los diversos módulos y componentes se relacionarán entre sí para cumplir con los requisitos del sistema.” (Bass, 2013)

La importancia de una arquitectura de software sólida radica en su capacidad para facilitar la gestión de la complejidad del desarrollo de software. Al dividir un sistema en módulos y capas bien definidos, se simplifica la tarea de desarrollar, mantener y escalar la aplicación a medida que evoluciona con el tiempo. Además, una arquitectura bien diseñada puede mejorar la flexibilidad y la reutilización de código, permitiendo a los desarrolladores agregar nuevas funcionalidades o realizar cambios sin afectar otras partes del sistema. En última instancia, una arquitectura de software sólida es fundamental para garantizar la calidad, la eficiencia y la escalabilidad de una aplicación en el largo plazo.

2.1.5 Arquitecturas más usadas en bancos

Las arquitecturas de software en los bancos tienen el desafío de adaptarse a las demandas del mercado, la regulación, la innovación y la seguridad. Los bancos pueden adoptar diferentes modelos de arquitectura según sus actividades.

2.1.6 Modularidad y calidad de código.

Es de suma importancia que la arquitectura de una solución cumpla ciertos requisitos. La modularidad y la seguridad son algunas de los motivos por lo que se debe diseñar la organización de las aplicaciones con rigor. Una buena arquitectura mejorará la eficiencia

de los programadores en el desarrollo, lo que significa también, beneficio financiero para quienes sean los dueños del proyecto. Meiappane en su investigación nos indica que: “Se debe realizar la evaluación de la arquitectura, de modo que los desarrolladores tengan la seguridad de que la arquitectura seleccionada reducirá el costo y el esfuerzo y también mejorará los diversos atributos de calidad como disponibilidad, reutilización, rendimiento, modificabilidad y extensibilidad. El éxito del sistema depende de la Evaluación de la Arquitectura.” (Meiappane, 2013)

La modularidad en software es la propiedad que permite dividir una aplicación en partes más pequeñas y autónomas, llamadas módulos, que tienen una función específica y se pueden reutilizar en diferentes contextos. La modularidad facilita el desarrollo, el mantenimiento y la comprensión del software, al reducir su complejidad y mejorar su estructura. Tiene varios beneficios para el software, como la reutilización del código, la mejora de la calidad, la simplificación en el proceso de pruebas, la adaptación al cambio, la integración con otras plataformas y la innovación tecnológica. La modularidad también favorece el trabajo en equipo, al permitir que cada módulo sea desarrollado por un grupo de personas independiente del resto.

2.1.7 Arquitectura de Microservicios

En arquitectura de software, se puede encontrar una gran cantidad de formas de diseño, sin embargo, en nuestro enfoque de servicios bancarios con soluciones web, podemos encontrar distintas filosofías de diseño de servicio. Las más comunes usadas en servicios bancarios web son:

- Arquitectura monolítica: Es en esencia, un servicio grande es el que maneja toda la aplicación. En esta forma, los distintos módulos tienen una dependencia unos de otros para el funcionamiento de la solución web.

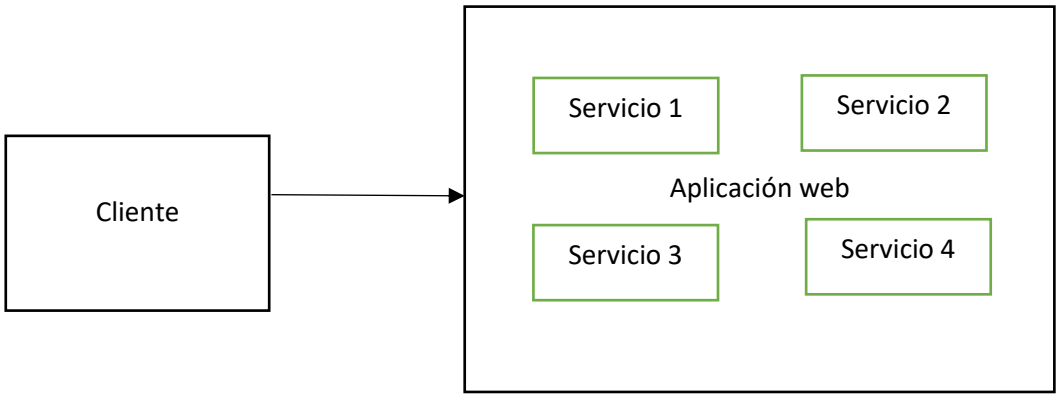

Ilustración 3 Arquitectura monolítica

- Arquitectura de microservicios: En esta forma de arquitectura, se enfoca en que la solución web esté conformado por varios servicios, a los que los que llamamos microservicios. Cada uno de estos servicios es capaz de manejar su propia lógica independientemente de otros módulos. Alkhodary nos dice que: “La arquitectura de microservicios es un estilo arquitectónico en el que una aplicación web se divide en varios servicios pequeños que se pueden compilar y enviar de forma independiente. Cada servicio está especializado para manejar parte de la lógica de las aplicaciones, y todos estos servicios trabajan juntos para cumplir el propósito de la aplicación web.” (Alkhodary, 2022) En este contexto, el diseño en microservicios tiene la ventaja de ser independientes y desacoplados, lo que quiere decir que esta arquitectura distribuida permite que cada microservicio sea desarrollado, implementado y escalado de manera independiente y por lo tanto, estos servicios residir en diferentes plataformas en la nube o servidores sin afectar su funcionamiento. Por ejemplo, podemos ver una arquitectura de microservicios como:

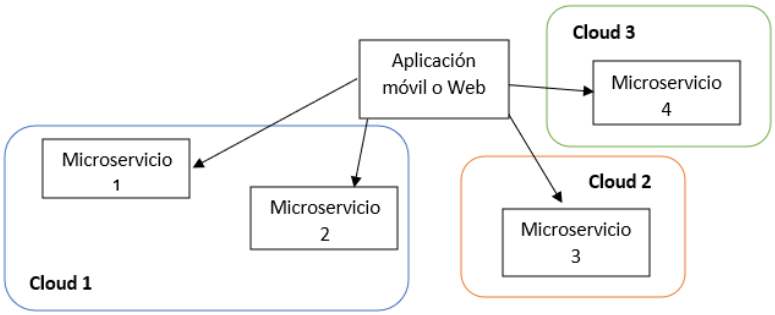

Ilustración 4 Arquitectura de microservicios

2.1.8 Backend For Frontend (BFF)

En este informe se pretende introducir al patrón de diseño BFF, para posteriormente hacer un análisis del uso de este en una arquitectura de solución web de instituciones financieras. Backend For Frontend es un patrón de arquitectura de software que propone crear un servicio intermedio entre el cliente y los servicios del backend, adaptado a las necesidades específicas de cada tipo de cliente. El objetivo de este patrón es reducir el acoplamiento entre el cliente y el backend, mejorar el rendimiento y la experiencia del usuario, y facilitar la evolución e innovación de ambos.

El autor, Samer Alkhodary, investiga los efectos del patrón BFF en la latencia, el uso de datos, las dependencias y el código del cliente en un entorno de microservicios. Se describe la metodología utilizada para la evaluación del patrón BFF en un entorno de microservicios (Alkhodary, 2022). Una herramienta de prueba utilizada estos microservicios BFF es Microusity, la cual realiza pruebas RESTful API, rastrea solicitudes procesadas por BFF, crea mapeo de solicitudes principales a sub-solicitudes y proporciona informes de prueba en formatos de visualización y texto para ayudar a los desarrolladores a rastrear y solucionar problemas (Rattanukul , 2023). Se pueden utilizar diferentes tipos de aplicaciones y requisitos que requieren microservicios, así como patrones para implementar almacenamiento de datos y devops en un entorno de microservicios (Brown, 2016). Una alternativa para mejorar la eficiencia de desarrollo y

ahorrar tiempo es el uso de una arquitectura estandarizada (Proof of Concept de BFFConnector) entre el frontend y el backend (Svensson, 2023).

BFF se aplica en entornos de microservicios, donde el backend se compone de varios servicios independientes y distribuidos que ofrecen una funcionalidad determinada. En lugar de que el cliente acceda directamente a estos servicios, lo hace a través de un BFF, que actúa como una fachada o un proxy, agregando, transformando y filtrando los datos según las necesidades del cliente. El BFF también puede implementar lógica de negocio específica para el cliente, como validaciones, autenticación o caché.

BFF se puede implementar usando diferentes tecnologías y protocolos, como REST, gRPC o GraphQL. Cada BFF pertenece al equipo responsable del cliente al que sirve, lo que permite una mayor autonomía y agilidad en el desarrollo. Cada cliente puede tener su propio BFF o compartirlo con otros clientes similares.

BFF tiene varias ventajas para el software, como la simplificación de la comunicación entre el cliente y el backend, la optimización del uso de recursos y datos, la personalización de la interfaz de programación y la mejora de la seguridad. BFF también facilita la adaptación al cambio, ya que permite modificar el cliente o el backend sin afectar al otro. Además, BFF favorece la innovación tecnológica, ya que permite incorporar nuevas funcionalidades o canales sin alterar la arquitectura existente. Según Rattanukul: “BFF es un patrón de microservicios que actúa como una interfaz entre el frontend y los microservicios de backend. Este patrón permite a los desarrolladores crear múltiples gateways para diferentes interfaces de usuario, como aplicaciones móviles y web, y ajustarlas a sus requisitos específicos. BFF es una solución para la complejidad de los sistemas de microservicios, ya que permite a los desarrolladores dividir la lógica de negocio en microservicios más pequeños y especializados. Además, BFF ayuda a mejorar

la escalabilidad y la seguridad de los sistemas de microservicios, ya que los desarrolladores pueden controlar el acceso a los microservicios de backend a través de la interfaz de BFF. En general, BFF es una solución útil para los desarrolladores que buscan una forma eficiente de integrar microservicios en diferentes interfaces de usuario." (Rattanukul, 2023)

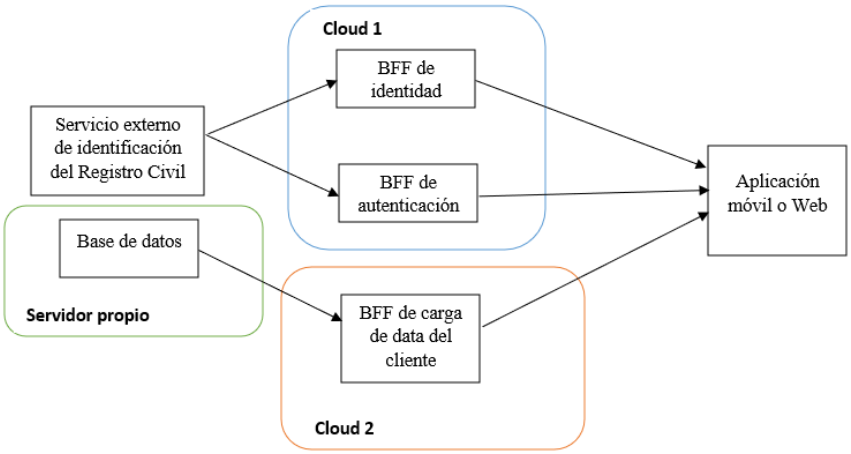

Ilustración 5 Ejemplo de arquitecturas con BFF

2.1.9 Comparativa económica de mantener activos los servicios.

Partiendo desde el punto de vista, en el que las instituciones financieras mantienen sus servicios en sus propios servidores por temas de seguridad, se hace un análisis de uso de recursos económicos entre los dos tipos de arquitecturas. Las aplicaciones monolíticas tienden a ser más eficientes, ya que toda la aplicación puede alojarse en un único servidor, simplificando la gestión y reduciendo los costos de infraestructura. En cambio, la arquitectura de microservicios puede implicar costos de infraestructura más elevados, ya que cada servicio independiente requiere su propio servidor, además aumenta la complejidad operativa y genera mayores gastos asociados a la infraestructura distribuida. A continuación, se muestra una tabla comparativa entre las dos arquitecturas:

Tabla 2 Comparación económica entre las dos arquitecturas

	Arquitectura Monolítica	**Arquitectura de Microservicios**
Mantenimiento de software	Costoso: "Los gastos de mantenimiento de las aplicaciones monolíticas aumentan con la evolución de los requisitos." (Click-IT, S.F)	Eficiente: "La coordinación inicial hace que el mantenimiento del código sea mucho más eficiente. Puede hacer cambios y encontrar errores más rápido. La reutilización del código también aumenta con el tiempo." (Click-IT, S.F)
Mantenimiento de infraestructura física	Rentable. "La arquitectura monolítica puede ser más rentable que la arquitectura de microservicios." (IBM, S.F)	En general, costoso. "Los microservicios pueden requerir una mayor inversión en infraestructura, monitoreo y gestión debido a la necesidad de mantener múltiples servicios." (IBM, S.F)

3. PROTOTIPO DE ARQUITECTURA CON BFF.

Para el desarrollo del presente proyecto fue necesario la implementación de un prototipo de una solución de software de una arquitectura que utilice BFF en el contexto de microservicios para la evaluación de esta. El enfoque de negocio del prototipo es hacia los servicios bancarios, por lo que se mostrará la solución con orientación a la banca.

3.1 Caso de uso de la implementación de BFF en prototipo de arquitectura.

Este caso de uso se refiere a la acción de un usuario de banca web al acceder y visualizar la página principal de su cuenta a través del prototipo de arquitectura con uso de BFFs. El prototipo proporciona una interfaz web que simula la página principal de la banca web, donde los usuarios pueden consultar información sobre sus cuentas, compras realizadas y pagos visualización de los servicios disponibles de uso. Para esto, es importante mencionar que, al ser un prototipo, el cliente de frontend solamente está recibiendo información de distintos microservicios y los coloca en el formato deseado en la pantalla principal. El desarrollo se lo hizo de forma local porque de esta forma tiene la ventaja de un ciclo de desarrollo más rápido porque se elimina la dependencia de la conectividad a la nube u otro tipo de servidores. Esto posibilita realizar cambios y pruebas rápidas, lo que agiliza las iteraciones durante las etapas de desarrollo. Además, esta aproximación evita la complejidad asociada con la configuración en la nube, la gestión de claves de

acceso y otros aspectos que podrían introducir complicaciones en el proceso de prototipado.

A continuación, se muestra el diseño de la arquitectura.

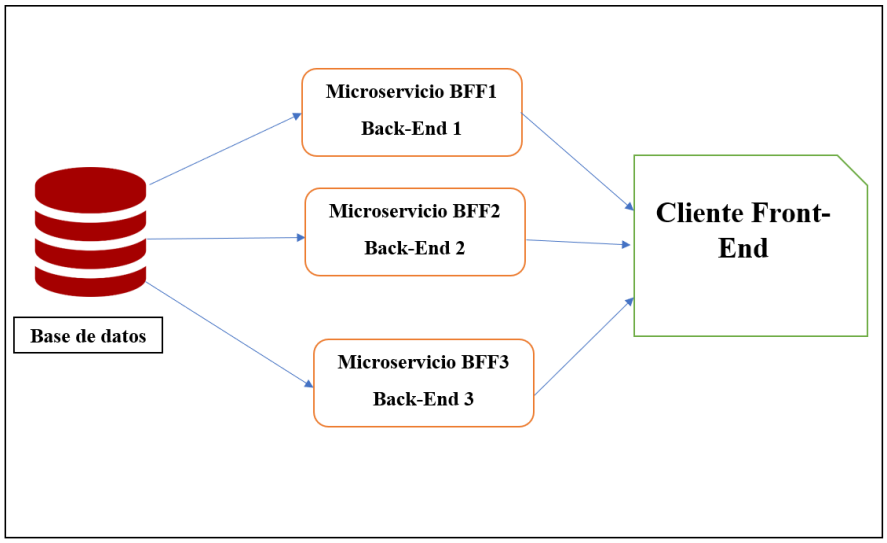

Ilustración 6 Arquitectura de prototipo

Para este caso propuesto, se muestra la forma en la que se intercambia información entre los distintos microservicios, proceso necesario para el funcionamiento de la solución.

1. El cliente de angular hace el request con el identificador entregado por el usuario.
2. El microservicio devuelve la información.
3. La data del cliente se guarda en el cliente de angular.
4. La información del cliente es compartida al módulo de compras para poder realizar el request.
5. El microservicio de compras responde con la información solicitada.
6. El modulo de servicios disponibles para el cliente tiene el mismo funcionamiento, que el de compras.

Se muestra un gráfico de como interactúan los microservicios:

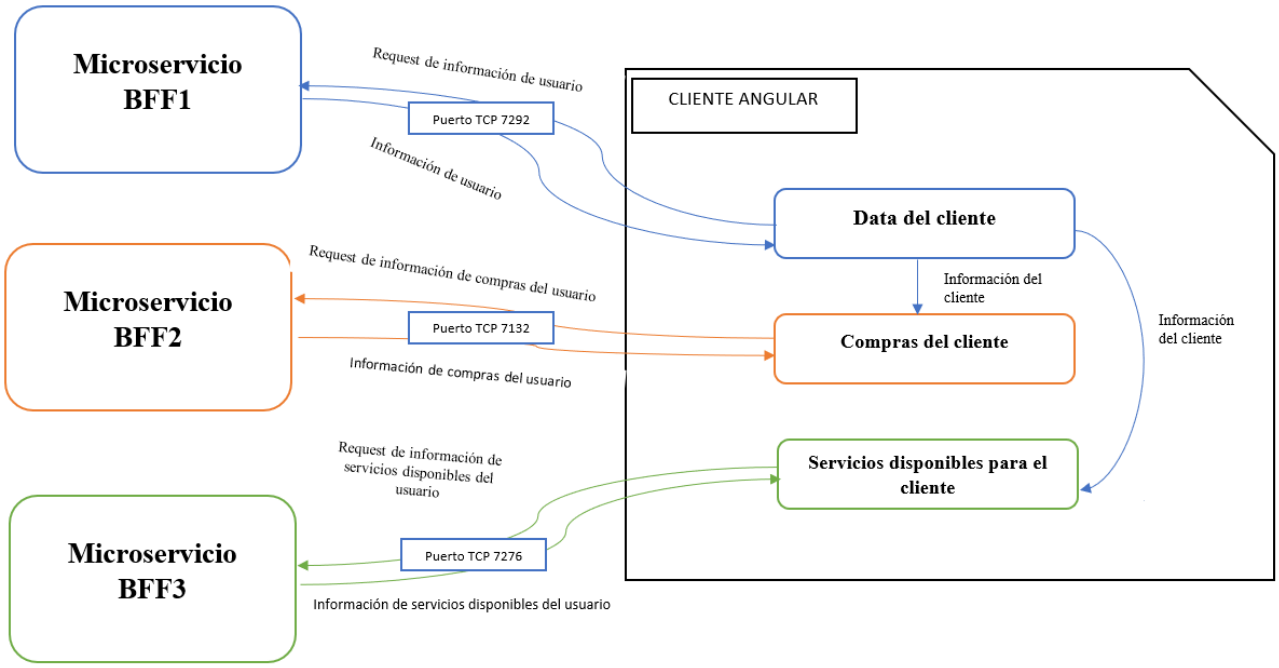

Ilustración 7 Forma de intercambio de información entre microservicios.

Como se puede ver, cada microservicio usa un puerto TCP diferente para establecer comunicación. Estos puertos actúan como puntos de entrada únicos que permiten a otros servicios o aplicaciones identificar y dirigirse específicamente a cada microservicio en el sistema. La asignación de puertos TCP individuales ayuda a gestionar las interacciones entre los diversos componentes de la arquitectura de microservicios, y con esto poder tener coexistencia y ejecución simultánea de múltiples servicios en la misma infraestructura.

Por otra parte, se configuración del proyecto hizo de forma manual de la siguiente forma:

```
builder.WebHost.ConfigureKestrel(options =>
{
    options.Listen(IPAddress.Parse("127.0.0.1"), 7292);
});
var app = builder.Build();
```

Ilustración 8 Configuración manual de la ruta.

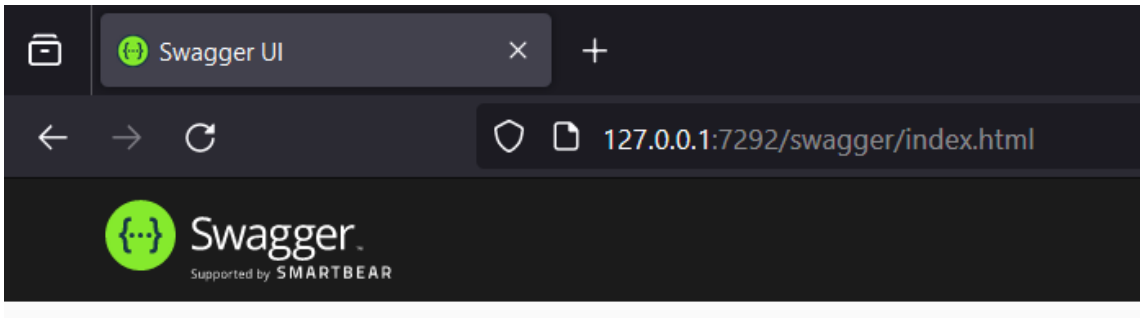

Ilustración 9 Pruebas del microservicio en swagger

Otra forma implementar los microservicios es a través de Docker, porque esta plataforma de contenerización ofrece un entorno aislado para cada microservicio, lo que garantiza consistencia en diferentes entornos. Además, tiene la capacidad de orquestar contenedores de manera eficiente lo que mejora la gestión de la infraestructura. Por otro lado, mantiene modularidad lo que ayuda a el despliegue de microservicios de forma individual. Además, AppMaster nos indica que: “Los contenedores creados con Docker son altamente portables, lo que permite a los desarrolladores mover aplicaciones entre diferentes entornos y plataformas fácilmente. Esta portabilidad garantiza que los microservicios puedan desplegarse y ejecutarse de forma coherente en diferentes sistemas, independientemente de la infraestructura subyacente” (AppMaster, 2023)

3.2 Problema que resuelve el diseño.

Normalmente, en soluciones web cuando se desarrolla una arquitectura robusta, al momento de fallos, la aplicación falla en su totalidad y no tiene el mínimo funcionamiento. Por esta razón, existen problemas en torno al negocio donde se apliquen este tipo de arquitecturas. Con el uso de BFFs en el contexto de microservicios, en el caso de que algún servicio falle, el resto de la aplicación podrá seguir funcionando, siempre y cuando el servicio que erre, no proporcione información para el resto.

En este contexto, el prototipo consume información de 3 BFFs distintos y se induce al fallo, por lo que en pantalla se visualizará los servicios en funcionamiento y los que fallen tendrán mensajes de error. Con esto notaremos que no toda la solución erra, si no que únicamente los componentes sin servicio. En el caso de que el microservicio que maneje otros falle, es decir el principal, toda la solución deja de tener funcionamiento.

3.3 Solución BFF.

Para la solución de BFF, se implementó código en Visual Studio 2022. Se creó un proyecto en blanco de ASP.NET Web API en el que están las herramientas necesarias para realizar las pruebas necesarias para la implementación en la arquitectura. Con esto se desarrolló una plantilla del BFF. Se agregó 3 proyectos extras en la solución para crear las distintas capas especializadas que caracterizan al BFF. Las capas de la solución son:

- Aplicación: La capa de aplicación es donde se encuentra toda la lógica del programa.
- Servicio: Es donde se desarrolla el contexto de servicios necesarios para el funcionamiento de la solución, en nuestro caso se encuentra la conexión de base de datos y la inyección de dependencia.

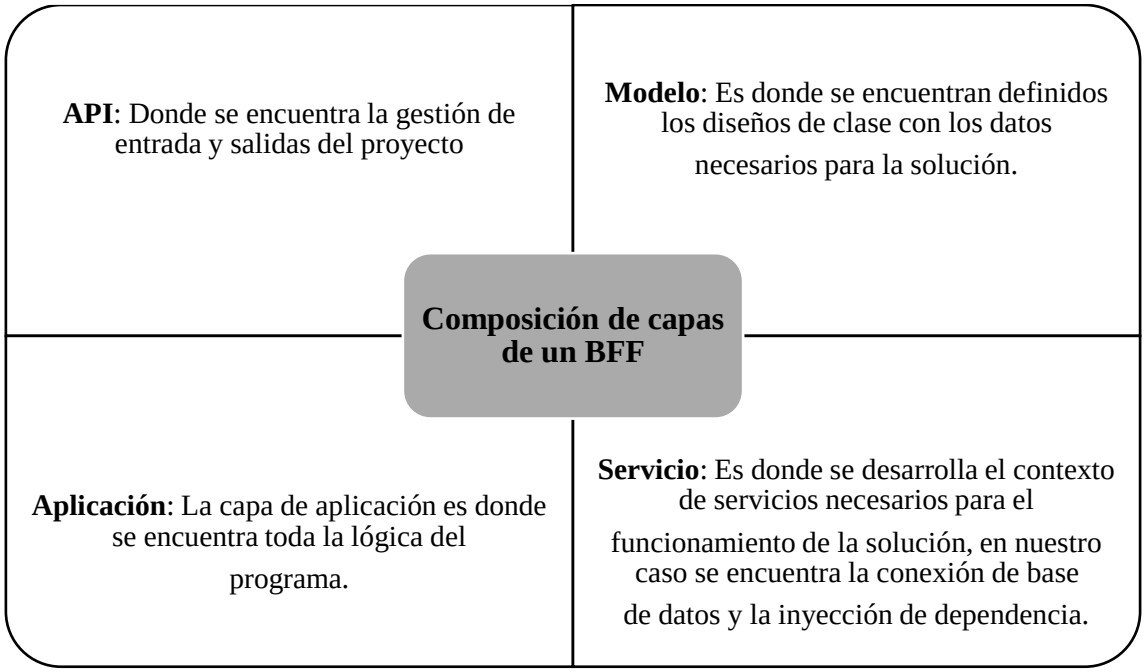

Ilustración 10 Composición en capas de BFF

Es importante entender las dependencias entre las capas que constituyen la arquitectura. En el siguiente gráfico, se visualizan las relaciones de dependencia clave. La capa de modelo se sitúa como la base fundamental, sin depender de ninguna otra capa, proporcionando así una independencia estructural. La capa de servicio, por su parte, se erige sobre el modelo, dependiendo exclusivamente de sus funcionalidades, información que contiene el modelado de las clases que estructuran la forma de recepción y entrega de la data. La capa de aplicación, al fusionar las capacidades de las capas modelo y servicio, establece dependencias con ambas para realizar sus operaciones específicas. Finalmente, la capa API, que actúa como la interfaz de exposición hacia el exterior, se vincula con las capas modelo y aplicación.

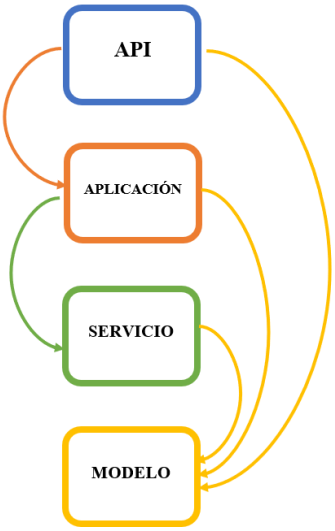

Ilustración 11 Dependencia entre capas de microservicios

3.3.1 Herramientas de Visual Studio usadas para el desarrollo.

Las herramientas usadas para el desarrollo del prototipo son:

- Visual Studio 2022: Es el ambiente de programación que más se adapta a las necesidades del proyecto.

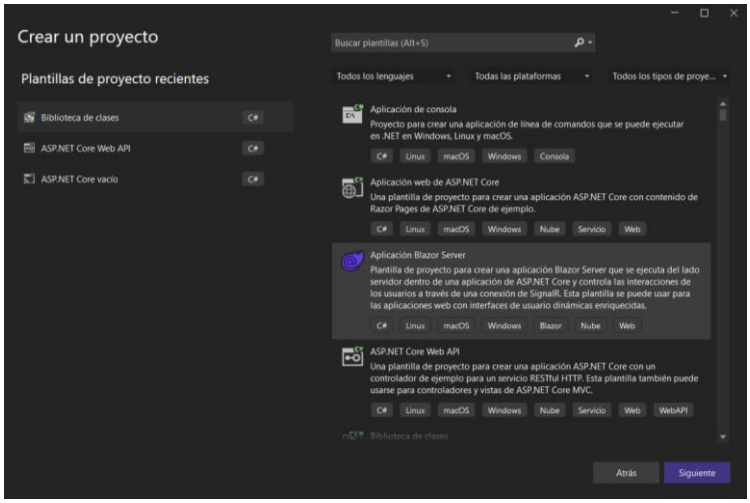

Ilustración 12 Creación del proyecto.

- Nugets o librerías: VS 2022 nos proporciona una gran cantidad de librererías listas para usar, las cuales se usaron:

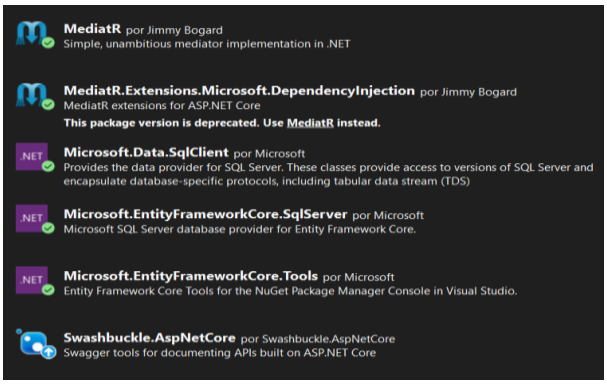

Ilustración 13 Nugets usados en la solución.

- Swagger: Es la herramienta incluida en VS2022 para las pruebas del API creado.

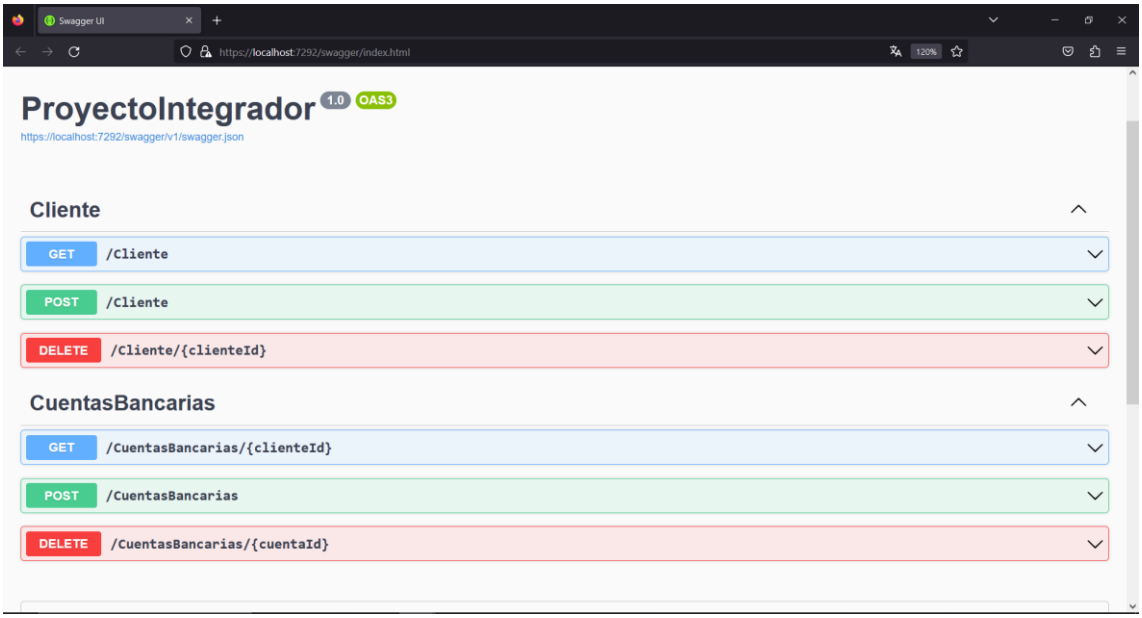

Ilustración 14 Swagger

- Microsoft SQL Server.

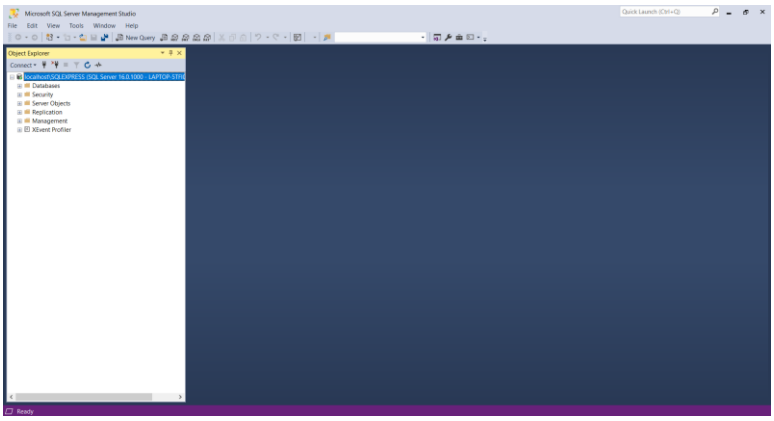

Ilustración 15 Servidor de base de datos.

3.4 Desarrollo.

Para el desarrollo de la solución, se tomó en cuenta que las herramientas sean compatibles entre sí, por este motivo la plataforma usada de todo el packtools es de Microsoft, tanto el ambiente de desarrollo, hasta el servidor de base de datos SQL.

3.4.1 Desarrollo SQL.

En el contexto de este proyecto, los scripts SQL son esenciales porque permiten la creación, gestión y manipulación de la base de datos que almacena los datos fundamentales para la aplicación. Además, facilitan la optimización de consultas, la implementación de lógica de negocio en procedimientos almacenados y funciones, y la realización de copias de seguridad, contribuyendo así a la eficiencia y el rendimiento del backend, que se encarga de consumir y procesar los datos desde la base de datos. Por lo que en la solución se desarrollaron scripts de:

- Creación de tablas relacionadas y no relacionadas.
- Storage procedures para el uso dentro de los BFF, en la consulta de datos y manipulación de tablas.

3.4.2 Desarrollo API.

Para el prototipo de la solución, los datos son consumidos desde una base de datos relacional SQL, por lo que es necesario usar algún

Las fases de desarrollo para la solución BFF son:

- Crear el proyecto nuevo.
- Crear las diferentes capas especializadas.
- Crear las dependencias necesarias entre las capas.
- Modelar las tablas de base de datos en la capa Modelo.

- Crear el contexto de conexión a base de datos en la capa de Servicio.
- Crear los interfaces necesarios con los métodos abstractos necesarios en la capa Modelo.
- Implementar las interfaces en los repositorios correspondientes a las tablas en la capa Servicio y desarrollar el cuerpo de los métodos.
- Desarrollar la lógica en la capa de Aplicación, se implementa el patrón de diseño MediatR para poder comunicar esta capa con la de API, sin tener una dependencia.
- Crear los controladores necesarios para manejar entradas y salidas del programa.

3.4.3 Frontend.

En el desarrollo de mi proyecto, también se en la creó el frontend utilizando tecnologías modernas. En este caso, se utilizó Angular un porque es una herramienta de trabajo de código abierto respaldado por Google, fue la elección principal para la construcción de la interfaz de usuario. La implementación de Angular, junto con TypeScript, proporcionó una estructura sólida y mantenible para el frontend, permitiendo una programación más orientada a objetos y mejorando la calidad del código. Además, para gestionar las solicitudes y respuestas a la APIRest, opté por la librería @angular/common/http de Angular. Esta librería facilitó la integración del consumo de servicios web en el frontend, permitiéndome realizar peticiones HTTP de manera eficiente y manejar las respuestas de la API de manera efectiva, contribuyendo así a la creación de una aplicación web dinámica.

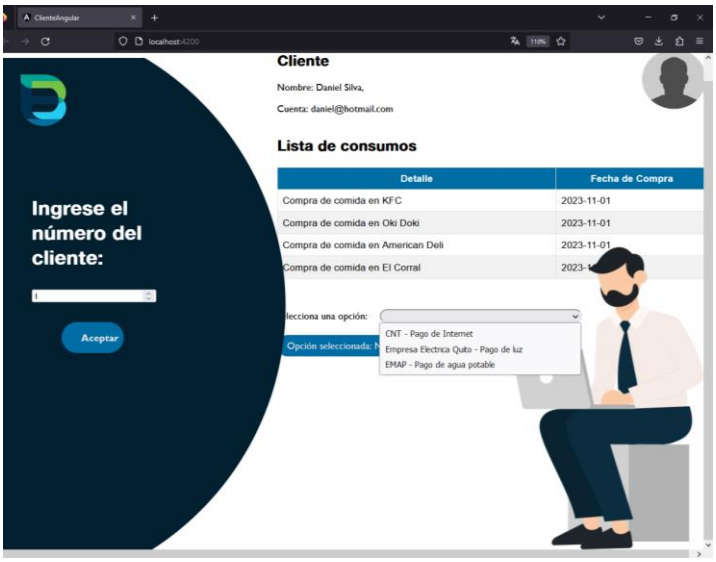

Ilustración 16 Pantalla de GUI

3.4.4 Pruebas de uso del prototipo.

Para esta parte del proyecto se realizaron pruebas de la aplicación. Se indujo al error al programa, de manera que se dejaron sin servicio a BFFs para que se note que no es necesario que un módulo de la solución falle, para que todo el contexto se quede sin servicio.

Las pruebas que se hicieron fueron:

- Únicamente BFF principal funcionando.
- Todos los BFFs en funcionamiento.
- BFF principal y un secundario en servicio.
- BFFs secundarios para ser consumidos.

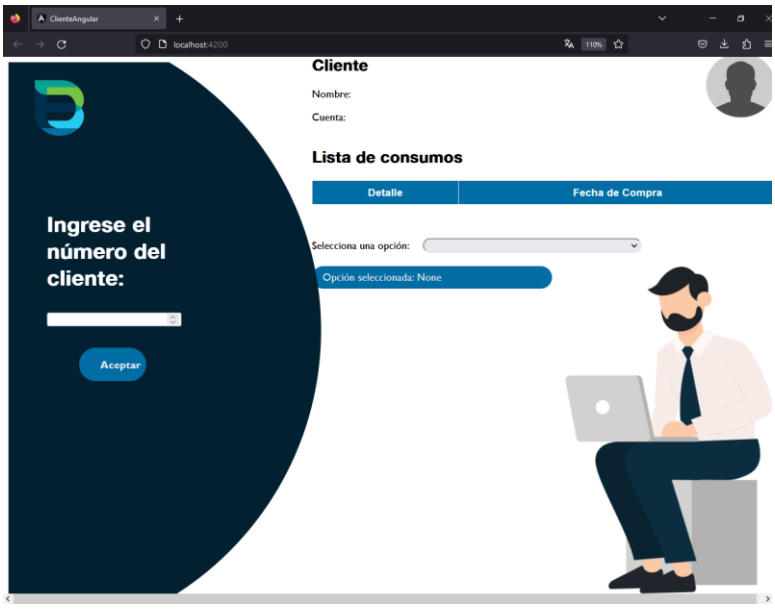

Ilustración 17 Pantalla inicial de aplicación.

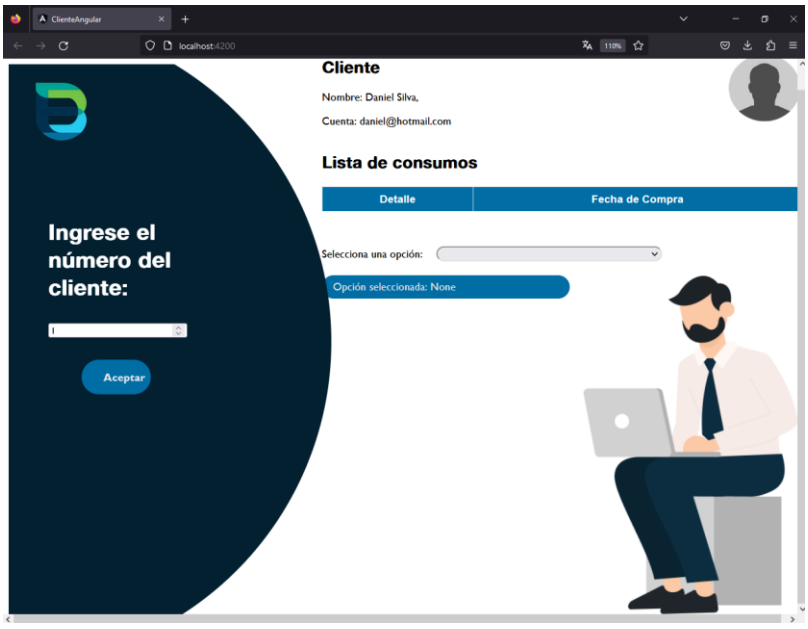

Ilustración 18 Un solo BFF en funcionamiento.

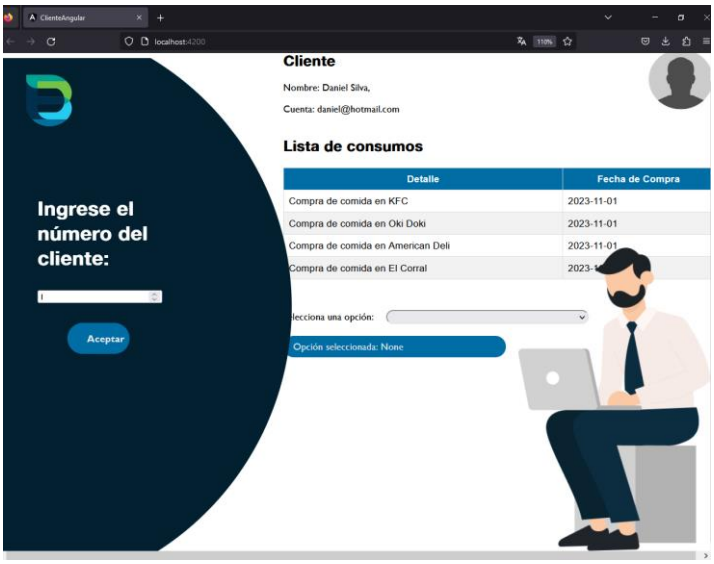

Ilustración 19 Dos BFFs en funcionamiento.

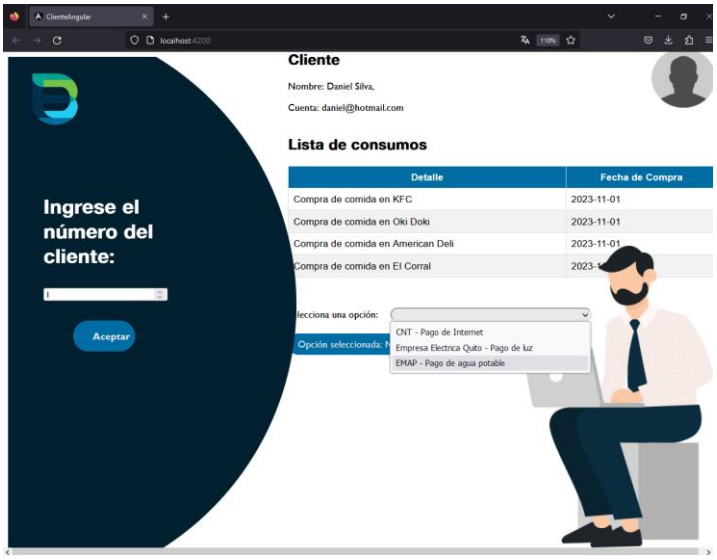

Ilustración 20 Tres BFFs en funcionamiento.

3.4.5 Resultados

El prototipo desarrollado fue sometido a evaluaciones con el objetivo de medir el comportamiento de la arquitectura implementada, especialmente en relación con los BFFs. Estas evaluaciones fueron diseñadas para analizar la eficacia y eficiencia de la comunicación entre el frontend desarrollado con Angular y TypeScript, y los distintos BFFs que actuaban como intermediarios entre la interfaz de usuario y los servicios backend. Posteriormente, se extrajeron conclusiones significativas de estas evaluaciones,

proporcionando una visión detallada del rendimiento general de la arquitectura con BFFs. Estas conclusiones no solo ayudaron a identificar áreas de mejora, sino que también sirvieron como base para la toma de decisiones informadas destinadas a optimizar la arquitectura y garantizar una experiencia de usuario óptima.

3.4.5.1 Resultados en el desarrollo.

La creación del scaffold de BFF lleva tiempo, porque es importante tomar decisiones de diseño de proyecto, y como interactúan las distintas capas entre ellas, aunque cada una de estas tengan una tarea o funcionalidad específica. Sin embargo, cuando el scaffold está listo, el desarrollo de los BFFs es sencilla y es fácil la creación de los microservicios, con esto todos los servicios tienen la misma estructura. En el sentido técnico, simplifica el desarrollo de soluciones más complejas, además lo que en el sentido económico nos dice que es más óptimo el uso de esta arquitectura para los trabajadores lo que mejora el rendimiento y la productividad, lo que se puede ver reflejado en beneficios económicos.

Para el prototipo, este es tiempo que tomó cada parte del prototipo en ser desarrollado.

Tabla 3 Tiempos de desarrollo por parte del prototipo.

Parte del desarrollo	Descripción	Tiempo (h)
1	Diseño y creación de scaffold BFF.	10
2	Creación de Scripts de Base de datos necesarios.	4
3	Desarrollo de BFFs para microservicio con scaffold.	10
4	Corrección de errores al entregar data.	1
5	Pruebas en Postman y Swagger del servicio.	1
6	Creación de Servicios en Angular.	3
7	Desarrollo de GUI HTML y CSS.	8
8	Corrección de errores de conexión con microservicios.	2
9	Pruebas de uso y casos de error.	1
10	Limpieza de código.	3
Total		43

En este sentido, con el motivo de poder exponer de mejor manera la comparación de tiempo de desarrollo del mismo proyecto, con distintas arquitecturas, se realizó una toma de datos de un proyecto realizado en una entidad financiera. Este proyecto fue migrado de una arquitectura monolítica a una que usa BFFs en el contexto de microservicios. La solución de software se realizó en partes y en sprints, y se encontró que las fracciones de desarrollo son muy similares entre ambas arquitecturas, pero el tiempo invertido en cada parte cambia, dependiendo la arquitectura utlizada. Por lo que podemos realizar la siguiente comparación.

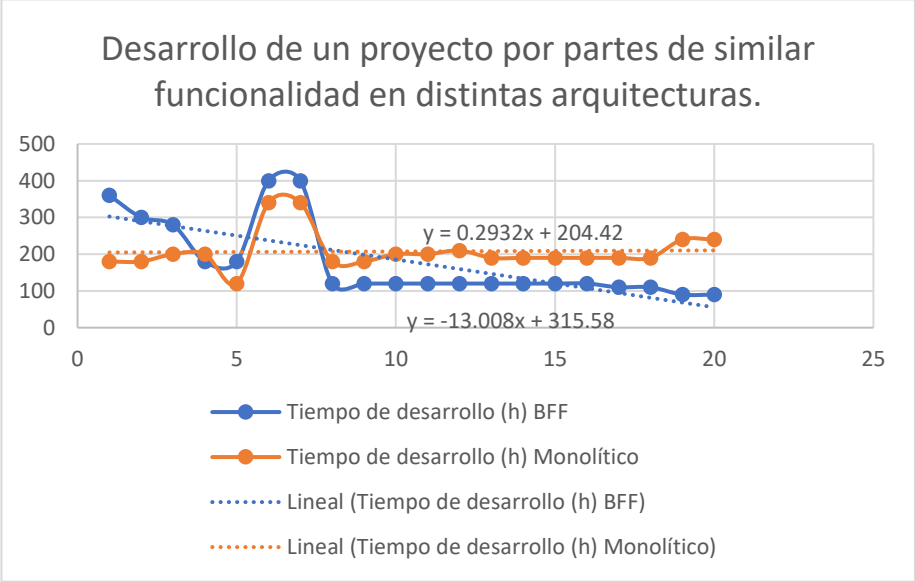

Ilustración 21 Comparación de tiempo de desarrollo BFF vs monolítica (elaboración propia, con datos de una institución financiera anónima).

En el gráfico anterior se distinguir que:

- De la parte 0 a la 5 es el desarrollo del backend de la solución.
- Del fragmento 5 al 10 el desarrollo del frontend.
- Y del 10 en adelante, son correcciones de bugs, mantenimientos e inclusión de nuevas funcionalidades.

En los resultados del experimento llevado a cabo, se observa una disparidad significativa en los tiempos de desarrollo entre la arquitectura que emplea el patrón de diseño BFF y

la arquitectura monolítica. En una fase inicial, el BFF exhibe un tiempo de desarrollo superior, posiblemente atribuible a la necesidad de establecer interfaces específicas para la comunicación entre el frontend y el backend. Sin embargo, a medida que el proyecto progresa y se enfoca en actividades de mantenimiento y actualización, se destaca una tendencia reveladora. La regresión lineal asociada con el BFF muestra una clara disminución en el tiempo necesario para estas tareas, sugiriendo una eficiencia creciente en comparación con la arquitectura monolítica. En contraste, la tendencia de desarrollo de la arquitectura monolítica parece ser inversa, indicando que a medida que el proyecto evoluciona, los tiempos de mantenimiento y actualización tienden a aumentar. Estos hallazgos sugieren que, a pesar de una inversión inicial más extensa, la implementación del BFF puede ofrecer ventajas sustanciales en eficiencia durante las fases subsiguientes del ciclo de vida del software.

En el tema económico, según ec.computrabajo un desarrollador de software en Ecuador gana una media de $5 la hora, y además, se asume que en cada grupo de programadores hay 6 personas. En el caso de que sean desarrolladores de planta en una empresa se hizo el cálculo con una media de 850 dólares al mes. Por lo que el desarrollo del proyecto, con los datos de tiempo por parte, se puede visualizar la diferencia económica:

Tabla 4 Comparación de tiempo y costos de desarrollo.

	BFF	**Arquitectura monolítica**
Tiempo de desarrollo solución (h)	2460	2120

Costo de desarrollo de toda la solución con 6 programadores por horas (Dólares)	73800	63600
Costo de desarrollo de toda la solución con 6 programadores de planta (Dólares)	78412.5	67575
Costo para mantener y actualizar en 10 partes de desarrollo con 6 programadores por hora (Dólares)	33600	60900
Costo para mantener y actualizar en 10 partes de desarrollo con 6 programadores de planta (Dólares)	35700	64706.25

3.4.5.2 Resultados en las pruebas.

Lo conexión del frontend con los microservicios es rápida y en el caso del prototipo, con información sencilla es óptima y genera una buena sensación como usuario. Al momento de la inducción al fallo, la arquitectura respondió de la manera esperada: si el BFF principal funciona, el resto de los BFF pueden no estar en servicio, pero el programa sigue

activo y devolviendo la información que esté disponible en el momento. Obviamente, si el microservicio principal no funciona, el resto de la solución se queda sin servicio también por la dependencia que tienen de este.

Las pruebas del prototipo se realizaron en un contexto local. Los experimentos locales realizados son importantes para la evaluación del rendimiento de los microservicios al cliente, proporcionando un entorno controlado y ágil para validar la eficacia de la implementación. En este caso, después pruebas en un entorno local, se facilita la detección temprana de errores. Además, proporcionan una visión inmediata del comportamiento del microservicio frente a cargas específicas de volumen de datos, esto es muy importante durante la fase de desarrollo y afinamiento del código. Por otro lado, el entorno local ofrece la oportunidad de evaluar la eficiencia del código y la interacción entre componentes sin depender de la complejidad de un entorno de producción. Sin embargo, hay que tomar en cuenta que las pruebas locales pueden no muestren completamente las condiciones de un ambiente de producción de alto tráfico de clientes, pero son muy importantes para identificar cuellos de botella en la aplicación, y comprobar la estabilidad de los microservicios.

Se realizaron pruebas de respuesta de la llamada desde el cliente de Angular hacia el microservicio principal. El servidor en el que se realizaron las pruebas estaba enfocado en maximizar el rendimiento de la aplicación, por lo que los únicos servicios activos fueron los microservicios, la base de datos y el cliente. Se hizo con una carga de 20000 datos para posicionar en pantalla, se tomaron los datos y los resultados fueron:

Tabla 5 Tiempos de respuesta de microservicio principal.

Request desde frontend	**Tiempo de respuesta de BFF (ms)**
1	102
2	98
3	73
4	206
5	104
6	79
7	97
8	93
9	61
10	114
11	60
12	77
13	103
14	65
15	98
16	73
17	82
18	73
19	72
20	96
21	75
22	82
23	107
24	68
25	80
26	81
27	68
28	84
29	68
30	84
31	67
32	59
33	61
34	80
35	62
36	65
37	60
38	87
39	73
40	70

Promedio	82.68

En el marco de la evaluación del tiempo de respuesta de nuestro microservicio, que se encarga de recuperar y procesar un conjunto de datos significativo de 20,000 elementos, los resultados obtenidos revelan un rendimiento extraordinariamente ágil, con un tiempo promedio de respuesta de tan solo 82.68 ms. Esta cifra muestra la eficacia y la capacidad de respuesta sobresaliente de nuestro microservicio al manejar volúmenes de datos considerables. La rapidez con la que se procesa este conjunto de datos no solo garantiza una experiencia de usuario optimizada, sino que también subraya la solidez y la eficiencia de nuestra infraestructura para hacer frente a cargas de trabajo sustanciales. Estos resultados positivos respaldan con firmeza la confiabilidad y el desempeño de nuestro microservicio, consolidándolo como un componente esencial dentro de nuestra arquitectura de sistemas.

4. CONCLUSIONES

La implementación de BFFs en el contexto de microservicios en el sistema bancario puede ofrecer mejoras sustanciales en la experiencia del cliente. Al utilizar BFF, se puede adaptar la lógica del backend específicamente para las necesidades del frontend, lo que resulta en una interfaz de usuario más ágil y eficiente. Además, la arquitectura de microservicios con BFFs facilita la modularidad y la escalabilidad, lo que permite a los bancos descomponer sus sistemas en componentes independientes y especializados. Esto no solo acelera el desarrollo y despliegue de nuevas funciones, sino que también mejora la flexibilidad y la capacidad de adaptación a cambios en el mercado financiero. La capacidad de escalar cada microservicio por separado garantiza un rendimiento óptimo, incluso en momentos de alta demanda. En conjunto, la implementación de BFF y microservicios en el ámbito bancario crea una infraestructura más ágil, adaptativa y

centrada en el cliente, mejorando significativamente la experiencia general del usuario. No solo mejora la resiliencia y la experiencia del usuario, sino que también facilita significativamente el mantenimiento continuo de las aplicaciones. Al dividir el sistema en servicios independientes, cada uno con su propia lógica de negocio, los equipos de desarrollo pueden realizar actualizaciones y mejoras en áreas específicas sin afectar el conjunto del sistema. Esta modularidad simplifica el proceso de mantenimiento, ya que los cambios pueden implementarse de manera más rápida y eficiente en comparación con las arquitecturas monolíticas.

Además, la independencia de los microservicios permite que los equipos de desarrollo trabajen de manera más autónoma en la evolución de sus servicios respectivos. Esto agiliza la implementación de nuevas funcionalidades, la corrección de errores y la optimización del rendimiento. Cada microservicio puede ser desarrollado, probado y desplegado de forma independiente, lo que facilita la gestión y actualización continua de la aplicación bancaria en su conjunto.

La adopción de la arquitectura de microservicios en el contexto bancario también aporta resiliencia y continuidad del servicio. Al descomponer las funcionalidades en servicios más pequeños e independientes, si un servicio específico llegara a experimentar una interrupción o dejar de funcionar, el resto de la solución puede seguir operando. Esta capacidad de aislamiento de fallos evita que un problema en un área específica afecte a la totalidad del sistema, garantizando una mayor estabilidad y disponibilidad para los usuarios. Además, la redundancia inherente en la distribución de microservicios contribuye a reducir el impacto de posibles fallos, ya que otros servicios pueden asumir la carga de trabajo, minimizando las interrupciones y asegurando la continuidad de los servicios críticos para la experiencia del cliente en el ámbito bancario. Este enfoque

modular y descentralizado no solo mejora la resistencia a fallos, sino que también facilita la detección y corrección rápida de problemas, optimizando así la confiabilidad general del sistema.

Además, la implementación de microservicios no solo brinda resiliencia técnica, sino que también mejora la sensación del usuario en caso de fallos. Al aislar los servicios, los impactos de una interrupción son localizados y controlados, lo que permite a los usuarios continuar utilizando otras funcionalidades sin verse afectados por el fallo de un componente específico. Esta capacidad de mantener operativas las partes no afectadas de la aplicación contribuye a minimizar la percepción de interrupciones por parte de los usuarios, mejorando así su experiencia. Además, la rápida detección y recuperación de fallos inherentes a la arquitectura de microservicios aseguran tiempos de inactividad reducidos, lo que, a su vez, fortalece la confianza del cliente en la estabilidad y confiabilidad del sistema bancario en su conjunto.

En el sentido económico, la mayor agilidad en el desarrollo y despliegue de nuevas funcionalidades gracias a la arquitectura de microservicios puede tener un impacto directo en la capacidad del banco para responder rápidamente a las demandas cambiantes del mercado y a las expectativas de los clientes. La rapidez en la introducción de servicios innovadores puede generar oportunidades para atraer y retener clientes, lo que se traduce en un potencial aumento de los ingresos. Además que, puede reducir los costos asociados con tiempos de inactividad no planificados. La capacidad de aislar y gestionar eficientemente fallos en servicios específicos minimiza el impacto económico de interrupciones del sistema. En distintos países, dejar sin servicio de banca electrónica a sus clientes conlleva multas y sanciones económicas. El desarrollo de una arquitectura de microservicios disminuiría este riesgo al componerse de módulos de distinta funcionalidad.

Como se pudo observar en la ilustración 17, tenemos dos tendencias de tiempo de desarrollo distintas para las dos arquitecturas evaluadas. En el caso del diseño que usa BFF en microservicios, aunque al inicio tarda un más su construcción, con el tiempo empieza a disminuir, como podemos ver cuando se hace una regresión lineal, tenemos como ecuación de la recta: -13.008x + 315.58. Lo que nos dice que la tendencia es de bajar el tiempo conforme pasa el tiempo. Sin embargo, para la arquitectura monolítica vemos la ecuación: y = 0.2932x + 204.42. Que lo que nos dice que conforme pasen partes de desarrollo de la solución, su tendencia será mínimamente a crecer. Lo que, como pudimos ver se traduce a pérdidas económicas para la institución.

Por otra parte, los tiempos de respuesta, con un promedio de 82.68 ms, obtenidos durante la evaluación de la solución, demuestran la capacidad excepcional de la arquitectura con BFF para gestionar eficientemente grandes conjuntos de datos. Estos resultados no solo subrayan la eficiencia operativa del servicio, sino que aseguran una gran experiencia de usuario. La agilidad demostrada en el manejo de volúmenes considerables de datos comprueba la capacidad del diseño.

5. RECOMENDACIONES

Es importante llevar a cabo pruebas exhaustivas de la arquitectura de microservicios y BFF en entornos que representen soluciones más robustas y con flujos de datos más complejos. La implementación de estas arquitecturas puede mostrar su verdadero potencial en situaciones donde la carga de trabajo es más intensa y los flujos de datos son más diversos. Al realizar pruebas en escenarios más desafiantes, se pueden identificar posibles cuellos de botella, evaluar la escalabilidad y asegurar que la arquitectura sea capaz de manejar eficientemente la complejidad inherente a soluciones bancarias de gran envergadura. Estos escenarios de prueba más exigentes permitirán ajustar y optimizar la arquitectura, garantizando su rendimiento óptimo y su capacidad para satisfacer las

demandas de soluciones bancarias completas y robustas. Los costos de la implementación y mantenimiento de otras infraestructuras con la infraestructura de microservicios con BFF puede depender de varios factores como la complejidad de la aplicación, las herramientas utilizadas, el personal disponible para el desarrollo, entre otros. Sin embargo, haciendo una evaluación con condiciones similares, en muchas veces resulta más conveniente el uso de microservicios, ya que el esfuerzo de desarrollo para los procesos de mantenimiento es menor, por ejemplo para manejar procesos de transaccionalidad de pagos de clientes, por la división de tareas y funciones en diferentes módulos independientes para este proceso, a diferencia de otros tipos de infraestructuras.

La capacidad de realizar actualizaciones independientes en servicios específicos facilita la gestión continua y reduce el impacto de cambios en toda la aplicación. Aunque inicialmente puede requerir una inversión más significativa en la configuración y gestión de la infraestructura, a largo plazo, la flexibilidad y modularidad de los microservicios a resultarían con mejor eficiencia operativa que compensa los costos iniciales. Como se evidencia en el subcapítulo 3.9, en términos económicos, la sostenibilidad a largo plazo de la arquitectura de microservicios se muestra como más rentable. Esta conclusión se refuerza al observar la información presentada en la ilustración 19, donde se aprecia una tendencia negativa a largo plazo en los costos asociados a la arquitectura de microservicios. Estos resultados respaldan la idea de que, aunque la implementación inicial puede requerir una inversión adicional, los beneficios en eficiencia operativa y mantenimiento a lo largo del tiempo hacen que la arquitectura de microservicios sea una opción económicamente más sólida.

Con el objetivo de fortalecer la postura de seguridad de la infraestructura tecnológica, se recomienda la revisión de las leyes de protección de datos y una autenticación con tokens, las cuales permitan garantizar un nivel mínimo de protección de la app en desarrollo.

Estas pruebas, realizadas por profesionales de seguridad certificados, simularán ataques reales con el fin de identificar posibles vulnerabilidades y puntos débiles en el sistema. La implementación de pruebas de hacking ético proporcionará una visión detallada de las posibles amenazas y ayudará a cerrar cualquier brecha de seguridad identificada. Se sugiere llevar a cabo estas pruebas en intervalos regulares, especialmente después de implementar actualizaciones significativas en la infraestructura o en el software.

6. REFERENCIAS

Aguirre, C., Donayres, K., Huarache, M., Gutiérrez, M., & Gamarra, M. (2021). *IMPACTO DE LA TRANSFORMACIÓN DIGITAL EN EL DESEMPEÑO ORGANIZACIONAL DE EMPRESAS DEL SISTEMA FINANCIERO*. https://repositorio.esan.edu.pe/bitstream/handle/20.500.12640/2909/2021_ADYFI_21-2_06_TC.pdf?sequence=1

Alkhodary, S. (2022). *The Evaluation of Using Backend-For-Frontend in a Microservices Environment*. LUND UNIVERSITY.

Americas Market Intelligence. (2023). *Estado de los pagos digitales y el comercio electrónico en LatAm*. https://americasmi.com/insights/pagos-digitales-comercio-electronico-en-latam-estadisticas-analisis/

Brown, K., & Woolf, B. (2016). *Implementation Patterns of Microservices Architectures*.

Cajamarca, I. (Ed.). (2021). *Medios digitales tuvieron 69% de las transacciones hechas en el primer semestre*. La Republica. https://www.larepublica.co/finanzas/medios-digitales-registraron-69-de-las-transacciones-financieras-en-el-primer-semestre-3226808

Cuesta, C., Ruesta, M., Tuesta, D., & Urbiola, P. (2015). *The digital transformation of the banking*.

F, Diener, & M, Špaček. (2021). *Digital Transformation in Banking: A Managerial Perspective on Barriers to Change*. https://doi.org/10.3390/su13042032

Khanchel, H. (2019). *The Impact of Digital Transformation on Banking*. https://doi.org/10.5430/jbar.v8n2p20.

Meiappane, A., Chitra, B., & Venkataesan, P. (2013). *Evaluation of Software Architecture Quality Attribute for an Internet Banking System*. https://doi.org/10.22394/2070-8378-2018-20-1-60-63

Minin, A. (2018). *Digital transformation in the banking industry*. https://doi.org/10.22394/2070-8378-2018-20-1-60-63

Omarini, A. (2017). *The Digital Transformation in Banking and The Role of FinTechs in the New Financial Intermediation Scenario*.

Rattanukul, P. (2023). *Microusity: A testing tool for Backends for Frontends (BFF) Microservice Systems*. https://doi.org/10.1109/ICPC58990.2023.00021

Rodríguez, M. (2023). *La era de la banca digital en Ecuador*.

Statista Research Department. (2023). *Distribución de métodos de pago de comercio electrónico en América Latina durante la pandemia del coronavirus (COVID-19) en 2020*. https://es.statista.com/estadisticas/1292122/metodos-de-pago-transacciones-online-latinoamerica/

Svensson, M. (2023). *Back-end for Front-end Connector to Reduce Development Time*.

Bass, L. (2013). *Software in Practice*.

https://edisciplinas.usp.br/pluginfile.php/5922722/mod_resource/content/1/2013%20-%20Book%20-%20Bass%20%20Kazman-Software%20Architecture%20in%20Practice%20%281%29.pdf

IBM. (2022). *¿Qué es la transformación digital?* https://www.ibm.com/es-es/topics/digitaltransformation

FORBES. (2020). *Banca tradicional vs Fintech*. https://www.forbes.com.mx/nuestra-revista-banca-tradicional-vs-fintech/

ec.computrabajo. (2020). *Salarios de Desarrollador Programador*. https://ec.computrabajo.com/salarios/desarrollador-programador

Amazon Web Services. (s. f.). *Monolithic vs. Microservices Architecture*. https://aws.amazon.com/es/compare/the-difference-between-monolithic-and-microservices-architecture/

IBM. (s. f.). *¿Qué son los microservicios?* https://www.ibm.com/es-es/topics/microservices

Click-IT. (s. f.). *Arquitectura monolítica vs arquitectura de microservicios: ¿cuál debo elegir*? https://click-it.es/arquitectura-monolitica-vs-arquitectura-de-microservicios-cual-debo-elegir/

AppMaster. (2023). *Utilización de Docker para la arquitectura de microservicios*. https://appmaster.io/es/blog/arquitectura-de-microservicios-docker