INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE

SISTEMAS DISTRIBUIDOS

PRACTICA No. 10

DESPLIEGUE EN NUBES

ALUMNO

GÓMEZ GALVAN DIEGO YAEL

GRUPO

7CM1

PROFESOR

CARRETO ARELLANO CHADWICK

FECHA DE ENTREGA

26 DE JUNIO DE 2025

INDICE

ANTESCEDENTES

La ingeniería de software ha experimentado una evolución significativa desde sus orígenes, particularmente en los procesos de desarrollo, despliegue y mantenimiento de sistemas informáticos. En las primeras etapas de esta disciplina, las aplicaciones eran ejecutadas directamente sobre servidores físicos locales, lo cual implicaba una elevada inversión en infraestructura, un control operativo centralizado y una fuerte dependencia del personal técnico para la instalación, actualización y monitoreo continuo del sistema. Este enfoque tradicional, conocido como *on-premise*, presentaba limitaciones evidentes en cuanto a escalabilidad, disponibilidad, y adaptabilidad frente a cambios en la demanda o en las condiciones del mercado.

Con el avance de la tecnología, se introdujeron los modelos de virtualización, permitiendo abstraer el hardware físico mediante la ejecución de múltiples sistemas operativos sobre una misma máquina física. Esta evolución mejoró el aprovechamiento de los recursos computacionales, pero no resolvió completamente las limitaciones estructurales del modelo tradicional, especialmente en términos de costos y escalabilidad dinámica.

La transformación más disruptiva en este campo llegó con el surgimiento del paradigma de la computación en la nube conocido en inglés como “Cloud computing”. Este modelo permite a las organizaciones y desarrolladores acceder a recursos informáticos, como cómputo, almacenamiento, bases de datos, redes, entre otros recursos, de manera flexible y bajo demanda, a través de Internet, sin necesidad de mantener infraestructura física propia. Las plataformas líderes como Microsoft Azure, Amazon Web Services (AWS) y Google Cloud Platform (GCP) ofrecen entornos altamente disponibles, resilientes y escalables, con esquemas de pago por uso que optimizan la eficiencia económica.

En el marco de esta revolución tecnológica, también surgieron herramientas y metodologías que impulsaron la modernización del ciclo de vida del software. Una de las más influyentes es Docker, una plataforma que introdujo el concepto de contenedorización. Esta técnica consiste en encapsular una aplicación y todas sus dependencias en un contenedor ligero, portátil y autosuficiente, capaz de ejecutarse de manera consistente en cualquier entorno. A diferencia de las máquinas virtuales tradicionales, los contenedores son más rápidos de desplegar, consumen menos recursos y simplifican la distribución del software.

El uso de Docker facilita la creación de imágenes de contenedor, que representan un snapshot exacto del entorno de ejecución de la aplicación. Estas imágenes pueden ser almacenadas en repositorios como Docker Hub, permitiendo su distribución, versionado y reutilización. Esto resulta fundamental en entornos colaborativos, donde múltiples desarrolladores pueden trabajar sobre la misma base, garantizando coherencia y compatibilidad en todo el ciclo de desarrollo.

Complementando esta infraestructura, surge GitHub, una de las plataformas más utilizadas para el desarrollo colaborativo de software, el cual se basa en un sistema de control de versiones Git,

GitHub permite almacenar el historial completo de cambios en el código, gestionar ramas de desarrollo, revisar pull requests y colaborar de forma asincrónica entre equipos distribuidos. Su integración con herramientas de automatización permite implementar pipelines de Integración Continua (CI) y Entrega Continua (CD), donde cada cambio en el código puede disparar automáticamente procesos de compilación, pruebas, generación de contenedores y despliegue en ambientes de producción.

Esta sinergia entre GitHub, Docker y los servicios de nube como Azure, ha dado lugar a un nuevo paradigma de desarrollo: ágil, automatizado y altamente escalable. En particular, la capacidad de vincular un repositorio de GitHub con una imagen Docker en Azure Container Registry o con un App Service, permite mantener una cadena de entrega continua, donde cualquier modificación en el código se traduce automáticamente en una nueva versión funcional de la aplicación desplegada en la nube.

Las ventajas que nos proporciona este enfoque incluyen:

- ✓ Colaboración distribuida y eficiente, sin importar la ubicación geográfica del equipo.
- ✓ Control de versiones detallado, facilitando la trazabilidad, auditoría y reversión de cambios.
- ✓ Automatización del proceso de despliegue, reduciendo tiempos de entrega y errores humanos.
- ✓ Escalabilidad horizontal, gracias a la posibilidad de replicar contenedores ante aumentos de carga.
- ✓ Aislamiento y portabilidad, lo que garantiza que las aplicaciones funcionen igual en todos los entornos.
- ✓ Despliegue modular mediante microservicios, permitiendo actualizar componentes individuales sin afectar el sistema completo.

PLANTEAMIENTO DEL PROBLEMA

En la actualidad, muchas organizaciones, ya sean educativas, administrativas o comerciales, siguen operando con sistemas informáticos monolíticos y centralizados que se ejecutan en entornos locales o en servidores tradicionales. Este modelo representa múltiples limitaciones: requiere infraestructura costosa y rígida, dificulta el escalado, presenta riesgos de disponibilidad ante fallas del servidor y complica las actualizaciones, poniendo en riesgo la continuidad operativa.

En un entorno tecnológico que demanda cada vez más accesibilidad remota, disponibilidad continua y experiencias multiplataforma, mantener estos sistemas tradicionales se vuelve insostenible. La dependencia de redes locales o equipos específicos limita seriamente el acceso a los servicios desde otros dispositivos o ubicaciones, afectando la experiencia del usuario final.

Adicionalmente, este enfoque convencional dificulta la adopción de prácticas modernas de desarrollo como la integración y entrega continua (CI/CD), el uso de contenedores (Docker) y el despliegue automatizado. Sin estas prácticas, los equipos de desarrollo enfrentan procesos manuales propensos a errores, lentos y poco eficientes, lo que compromete la agilidad del negocio y retrasa la evolución tecnológica.

Por tanto, el problema radica en que la ausencia de una arquitectura basada en servicios desplegada en la nube impide a las organizaciones escalar, mantener y evolucionar sus sistemas de manera eficiente y sostenible. Esta falta de modernización limita su capacidad de respuesta ante nuevas demandas tecnológicas, restringe el acceso remoto, y dificulta la adopción de una estrategia de desarrollo ágil y robusta.

PROPUESTA DE SOLUCIÓN

Como respuesta a esta problemática, se propone la migración de los servicios web, microservicios y la interfaz de usuario PWA hacia un entorno de computación en la nube, adoptando una arquitectura moderna, escalable y eficiente. Esta migración permitirá superar las limitaciones de los sistemas tradicionales, brindando alta disponibilidad, acceso remoto, facilidad de mantenimiento y una experiencia de usuario mejorada.

La solución contempla el rediseño e implementación del sistema como un conjunto de microservicios independientes, cada uno responsable de una funcionalidad específica. Estos microservicios, serán contenedorizados utilizando Docker, asegurando portabilidad y consistencia entre ambientes de desarrollo, pruebas y producción. Las imágenes generadas se almacenarán y gestionarán desde Docker Hub.

Todo el ciclo de vida del software será gestionado a través de GitHub, plataforma que centraliza el código fuente, gestiona el control de versiones y facilita la integración y entrega continua (CI/CD). Esto permitirá automatizar la construcción, pruebas y despliegue de los servicios, garantizando trazabilidad, eficiencia y rapidez en la entrega de nuevas versiones.

Una vez desarrollados y contenedorizados, los servicios web, microservicios y la PWA serán migrados y desplegados en la nube, específicamente en las plataformas Microsoft Azure y Google Cloud Platform (GCP), utilizando servicios como Azure App Service, Azure Static Web Apps, así como Cloud Run, App Engine y el uso de buckets de almacenamiento estático (Google Cloud Storage) para hospedar sitios web estáticos. Esta infraestructura gestionada proporciona beneficios clave como escalabilidad automática, balanceo de carga, monitoreo integrado, seguridad, respaldo y eliminación de la dependencia de infraestructura física, asegurando además una alta disponibilidad y facilidad de administración para las aplicaciones desplegadas.

Este proceso de migración hacia la nube no solo optimiza el rendimiento y la disponibilidad del sistema, sino que también prepara la solución para responder a escenarios de alta demanda, movilidad, y crecimiento continuo, alineándola con las mejores prácticas de desarrollo y operación modernas.

MATERIALES Y METODOS

Para el desarrollo, contenedorización y despliegue del sistema, se utilizaron las siguientes herramientas tecnológicas:

Materiales

Para la realización de esta práctica de configuración básica de nubes públicas, se utilizaron los siguientes recursos:

- ✓ Lenguaje de programación:
  - Java: Para la implementación de lógica del backend
  - HTML, CSS y JavaScript: Para el desarrollo de las interfaces web
- ✓ **Frameworks y libreria:**
  - HTTPServer (Java nativo) para microservicios
  - Gson para manejo de datos en formato JSON
  - Service Worker, manifest.json y localStorage para funcionalidades PWA
- ✓ **Herramientas de desarrollo:**
  - Dokcer para la creación de contenedores que encapsulan los servicios con sus dependencias
  - Docker Hub como repositorio para almacenar las imágenes de los contendores
  - GitHub para el control de versiones y la automatización del ciclo de vida del Software (CI/CD)
  - PostgreSQL como sistema de gestión de bases de datos
- ✓ **Proveedor de nube:** Microsoft Azure y Google Cloud Platform (GCP)

  Plataforma en la que se desplegaron los servicios como aplicaciones web y aplicaciones web estáticas

DESARROLLO DE SOLUCIÓN

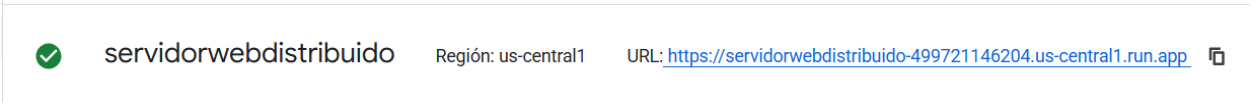
Google Cloud Platform (GCP)

La práctica inició con la preparación del servicio web, desarrollado en Java, que sería desplegado en la nube como contenedor. Para ello, se creó un archivo Dockerfile en el repositorio de GitHub correspondiente al servicio web. Este archivo contenía las instrucciones necesarias para construir la imagen del contenedor, incluyendo la copia del archivo .jar, la configuración del entorno y la instrucción de ejecución.

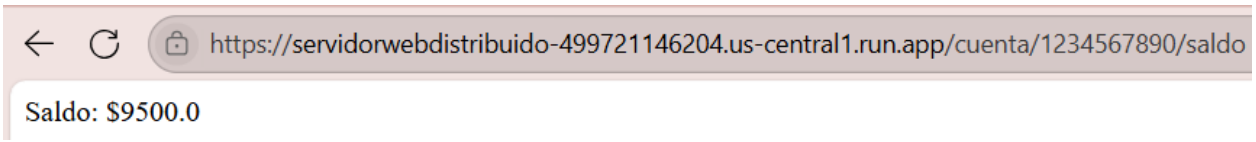
```
FROM maven:3.9.6-eclipse-temurin-21 AS build
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests

FROM eclipse-temurin:21-jdk
WORKDIR /app
COPY --from=build /app/target/sistemabancario-0.0.1-SNAPSHOT.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Una vez confirmado que el Dockerfile funcionaba correctamente en pruebas locales, se procedió a realizar la conexión entre GitHub y Google Cloud Platform mediante la herramienta Cloud Build. Esta herramienta de GCP permite crear integraciones automáticas con repositorios de código para compilar, construir imágenes y desplegar aplicaciones. Dentro de la consola de Cloud Build, se configuró el origen del repositorio, especificando la ubicación exacta del Dockerfile y habilitando el acceso mediante una conexión segura autenticada con GitHub.

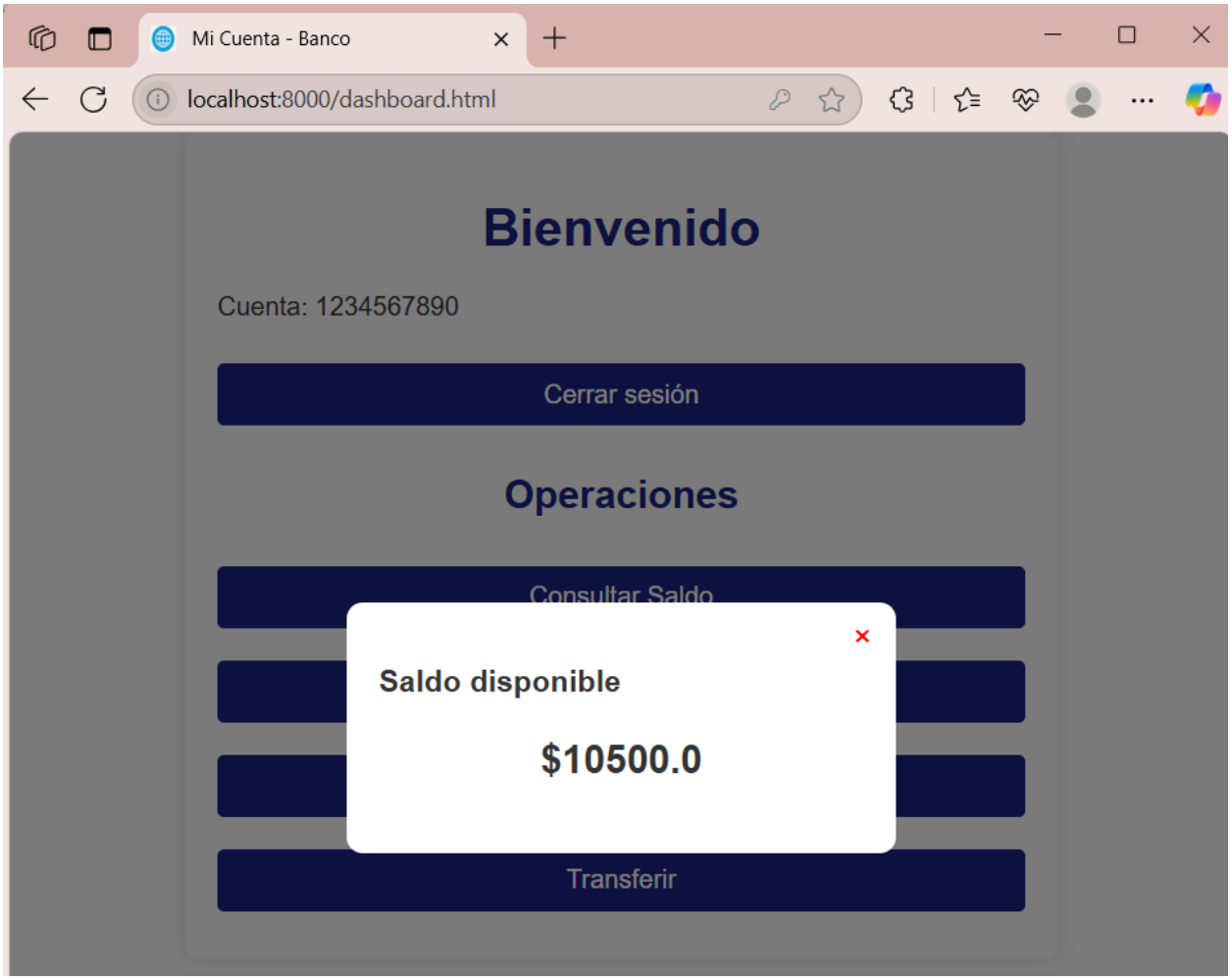
Al completar esta integración, se disparó automáticamente el proceso de construcción del contenedor: Cloud Build descargó el código fuente, ejecutó el Dockerfile y generó una imagen que fue desplegada mediante Cloud Run, un servicio serverless que permite ejecutar contenedores sin necesidad de administrar infraestructura. Al finalizar el despliegue, Cloud Run proporcionó una URL pública donde el servicio web quedó disponible.



Como siguiente paso, se realizó una prueba básica de funcionamiento. Se accedió a la URL pública del servicio para validar que respondiera correctamente a solicitudes HTTP.

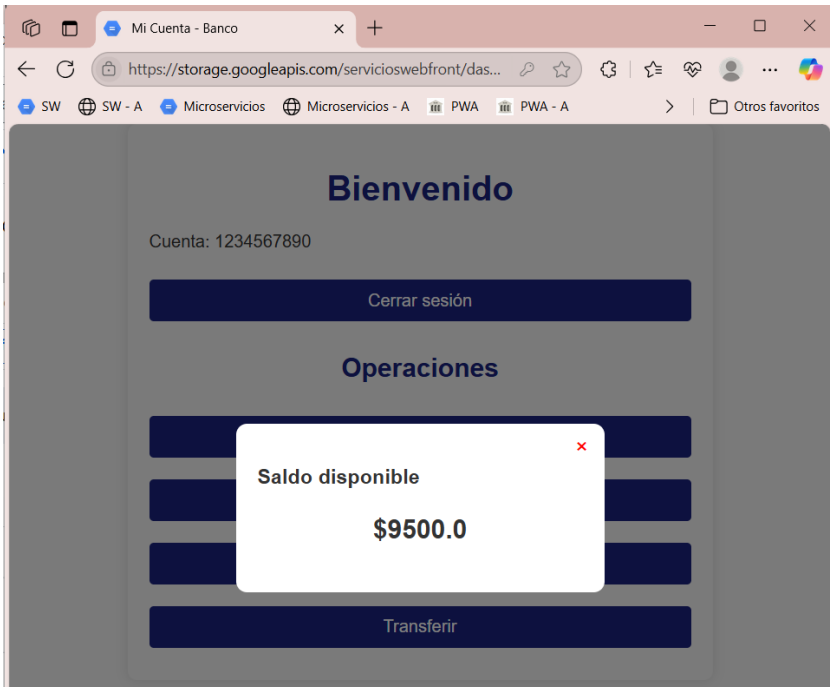


Posteriormente, se ejecutó localmente el frontend del sistema, configurado para conectarse al backend desplegado en Cloud Run. Esta conexión permitió validar que el servicio respondía adecuadamente a peticiones generadas desde el entorno de usuario.



Tras confirmar el correcto funcionamiento tanto del backend como del frontend en local, se procedió a desplegar también la interfaz gráfica como aplicación web en la nube. Para ello, se utilizó Google Cloud Storage, específicamente la funcionalidad de sitio web estático. En este proceso, se creó un bucket de almacenamiento configurado con acceso público y se cargaron en él todos los archivos del frontend: index.html, estilo.css, archivos JavaScript, el manifiesto PWA y el service worker. El bucket fue configurado para servir como página principal el archivo index.html

Finalmente, se accedió a la URL pública generada por el bucket para verificar que la interfaz estuviera operativa, conectada al backend previamente desplegado, y funcionando correctamente como una Progressive Web App (PWA).



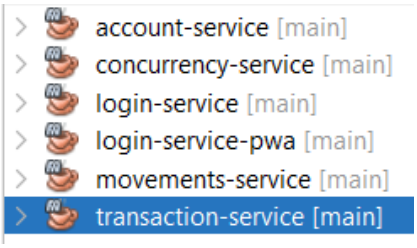
Enseguida continue con el despliegue de la práctica Microservicios. El primer paso para llevar a cabo el despliegue fue la creación del sistema de persistencia de datos. Para ello, se configuró una base de datos en la nube mediante Cloud SQL, el servicio gestionado de bases de datos de Google Cloud Platform (GCP). Se creó una instancia de PostgreSQL, especificando las credenciales de acceso, nombre de la base de datos y parámetros de conexión remota.

Una vez creada la instancia, se accedió a través de la terminal mediante el cliente de PostgreSQL (psql) y se ejecutaron los comandos necesarios para construir las tablas correspondientes al sistema: Cuentas, Movimientos y demás estructuras necesarias. Estas tablas fueron creadas asegurando claves primarias, relaciones foráneas y restricciones de integridad.

```
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 17.5)
WARNING: psql major version 16, server major version 17.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> \c banco
Password:
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 17.5)
WARNING: psql major version 16, server major version 17.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "banco" as user "postgres".
banco=>
```

Con la base de datos en funcionamiento, se procedió a modificar el código fuente de cada microservicio para establecer la conexión adecuada a PostgreSQL en la nube. Se actualizaron las cadenas de conexión en cada servicio (account-service, transaction-service, login-service, entre otros), asegurando la correcta autenticación y conexión hacia la base de datos remota. También se ajustaron los métodos de acceso a datos para que interactuaran directamente con la nueva instancia gestionada.

Posteriormente, se reorganizó la estructura del proyecto original, que estaba diseñado como una aplicación monolítica, dividiéndolo en proyectos separados. Cada microservicio fue configurado con su propia clase principal (main), dependencias necesarias y un archivo Dockerfile independiente que permitiera contenerizar su ejecución. Esta separación garantizó un diseño modular, escalable y mantenible.

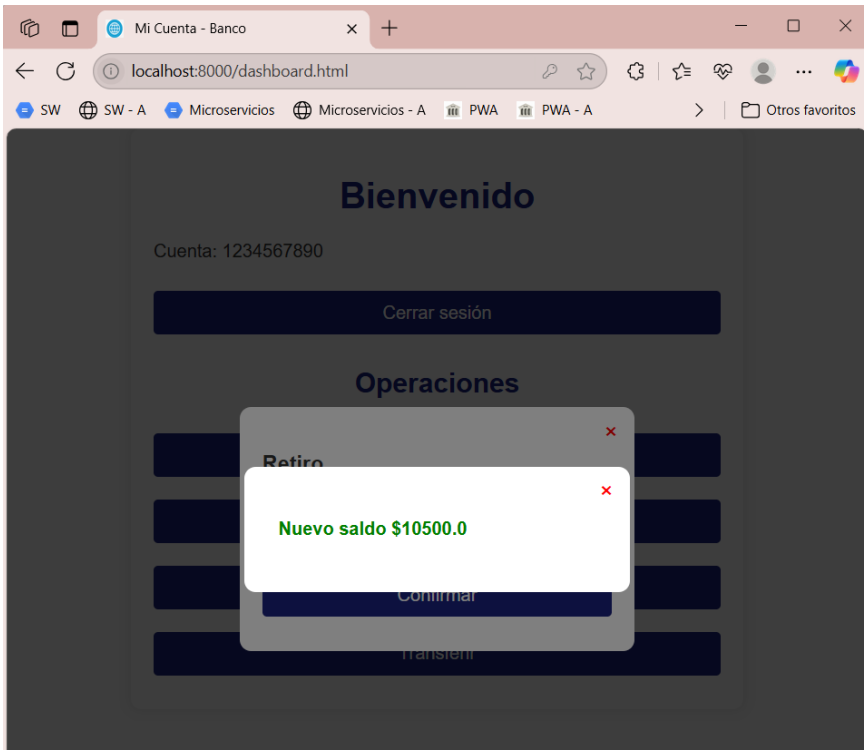


Para el proceso de despliegue se hizo uso de diversas herramientas clave proporcionadas por Google Cloud Platform (GCP). En primer lugar, GitHub fue utilizado como plataforma de control de versiones, donde se mantuvo el código fuente de cada microservicio, ya sea en un repositorio central o en subcarpetas independientes. Esto permitió una colaboración efectiva y trazabilidad de los cambios durante el desarrollo. A continuación, se empleó Cloud Build para automatizar la construcción de imágenes de contenedor. Se configuraron *triggers* específicos

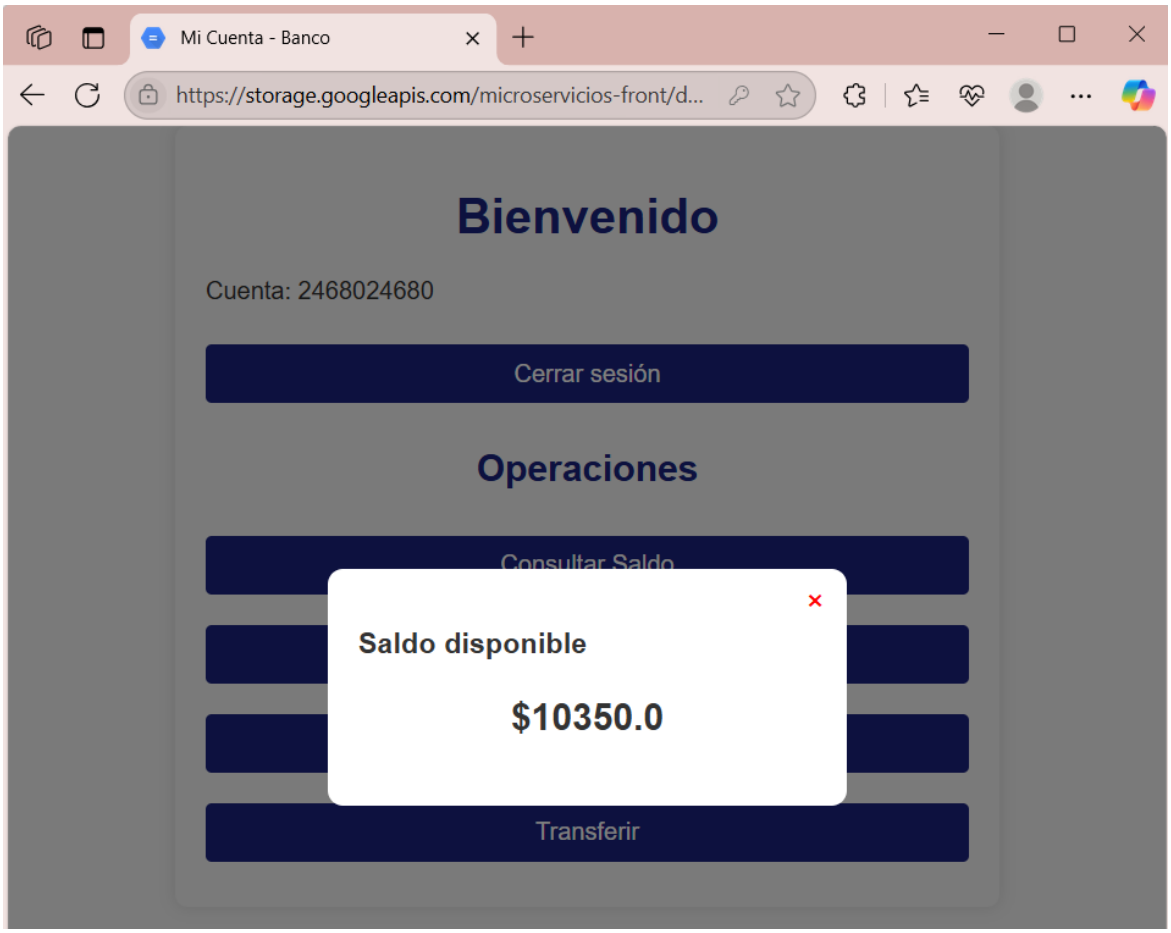
para cada microservicio, apuntando a la ubicación exacta del archivo Dockerfile dentro del repositorio. Cada vez que se realizaba un *push* o un despliegue manual, Cloud Build se encargaba de generar la imagen correspondiente del contenedor. Una vez construida la imagen, esta era desplegada mediante Cloud Run, el servicio serverless de GCP que permite ejecutar aplicaciones en contenedores sin necesidad de administrar servidores. Cada microservicio fue desplegado de forma individual en este entorno, recibiendo una URL pública que permitió su consumo y prueba desde el frontend o desde otras partes del sistema.

account-service	Repositorio	0	us-central1	Permitir sin autenticación	Todas	hace 2 días
concurrency-service	Repositorio	0	us-central1	Permitir sin	Todas	hace 2 días
login-service	Repositorio	0	us-central1	Permitir sin autenticación	Todas	hace 2 días
transaction-service	Repositorio	0	us-central1	Permitir sin autenticación	Todas	hace 2 días

Una vez desplegados, se actualizaron las URL en el frontend para que cada funcionalidad apuntara al microservicio correspondiente. Se realizaron pruebas locales para verificar la comunicación correcta entre la interfaz y los servicios.



Finalmente, el frontend completo fue desplegado como sitio web estático utilizando Google Cloud Storage. Se creó un bucket configurado como sitio público, en el cual se subieron todos los archivos del cliente: HTML, CSS, JS



Por último, se llevó a cabo el despliegue de la aplicación PWA (Progressive Web App), consolidando el ecosistema del sistema distribuido basado en microservicios. Debido a que el backend ya estaba estructurado en microservicios desde el despliegue anterior, fue posible reutilizar los servicios previamente desplegados, integrándolos ahora al flujo de la aplicación progresiva. No obstante, para cubrir las funcionalidades específicas de esta nueva interfaz, se desarrollaron dos microservicios adicionales.

El primero de ellos, denominado login-service-pwa, extendió las funcionalidades básicas de autenticación presentes en el microservicio original de inicio de sesión. En esta nueva versión, se incorporó la capacidad de mostrar un inicio personalizado, trayendo el nombre del usuario desde la base de datos tras una autenticación exitosa, con el objetivo de enriquecer la experiencia de usuario.

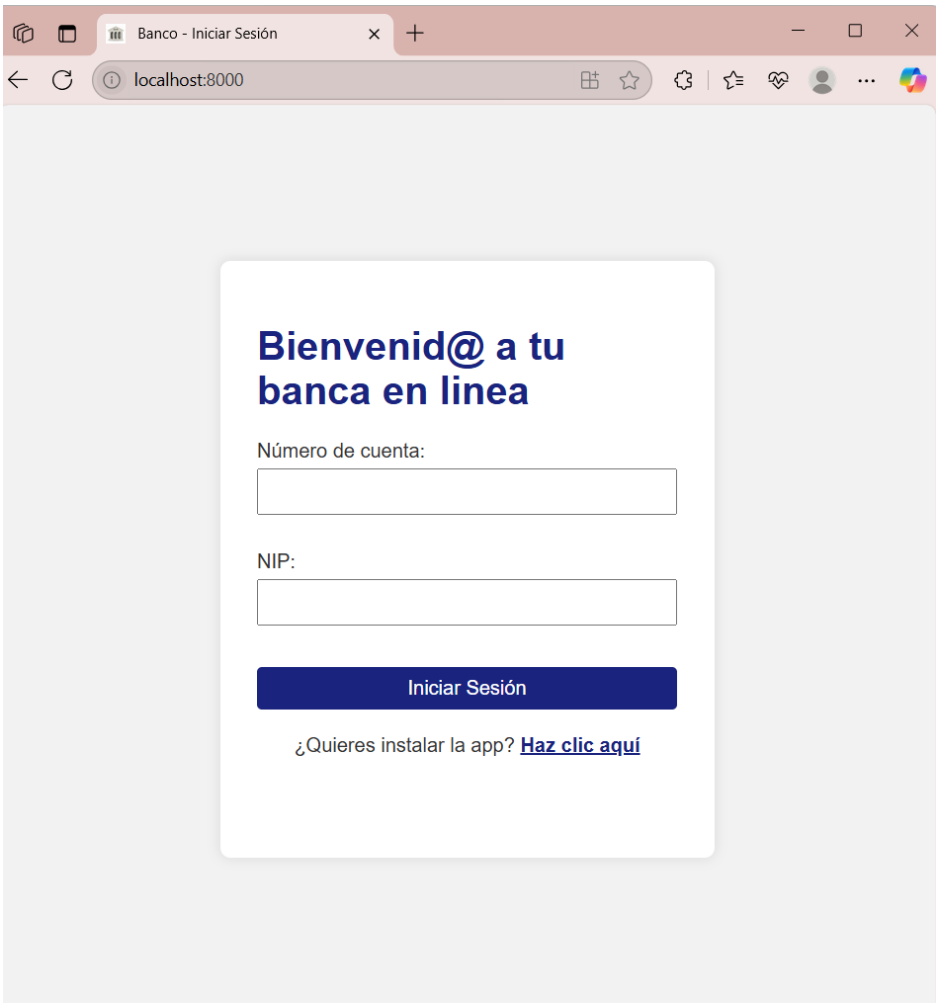
login-service-pwa	Repositorio	0	us-central1	Permitir sin autenticación	Todas	hace 2 días

El segundo microservicio implementado fue movements-service, cuya finalidad fue gestionar los movimientos bancarios del usuario. Este servicio permitió registrar, consultar y visualizar los

movimientos asociados a una cuenta específica, fortaleciendo así la parte funcional de operaciones del sistema.

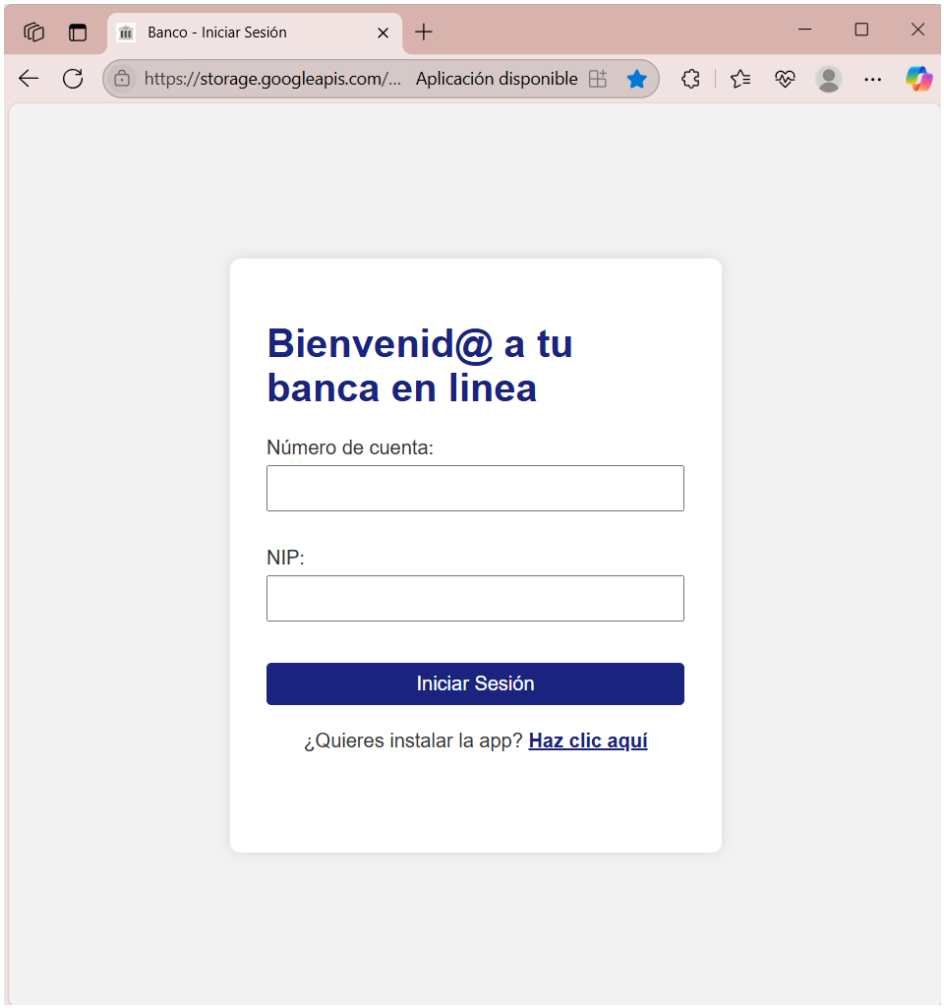
movementes-service	Repositorio	0	us-central1	Permitir sin autenticación	Todas	hace 2 días

Una vez desarrollados estos nuevos servicios, se procedió a actualizar el frontend de la PWA, incorporando las rutas y peticiones necesarias para interactuar con los microservicios correspondientes. Se realizaron pruebas locales exhaustivas para validar la correcta comunicación entre la interfaz PWA y los servicios backend, verificando el flujo completo de autenticación, visualización de datos y registro de movimientos.



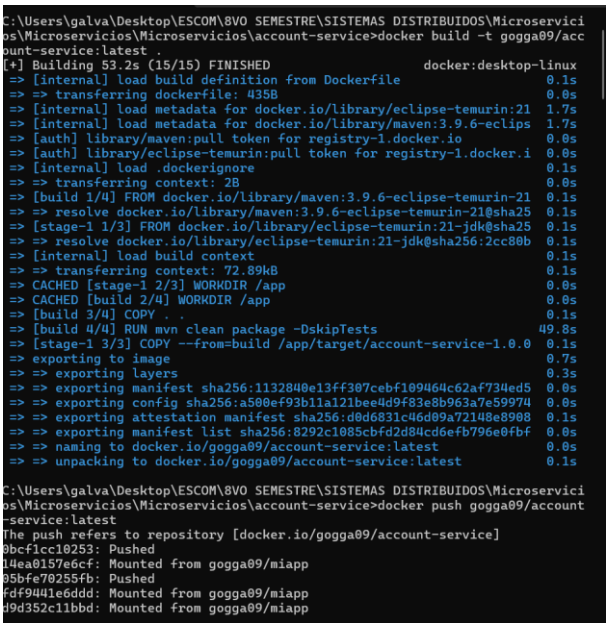
Finalmente, tras confirmar el funcionamiento correcto de toda la arquitectura, se procedió al despliegue de la PWA en Google Cloud Platform (GCP). Para ello, se utilizó Google Cloud Storage (GCS) como solución de hosting para sitios web estáticos, cargando en un *bucket* la carpeta que contenía el frontend compilado de la PWA. De esta forma, la aplicación quedó accesible desde cualquier dispositivo con navegador web, ofreciendo una experiencia fluida,

confiable y con capacidades nativas como instalación en el dispositivo, funcionamiento offline y notificaciones.



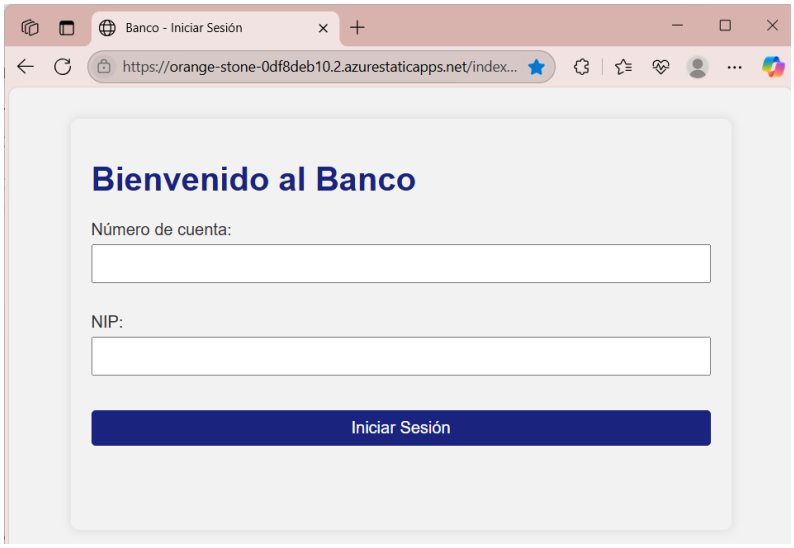
Microsoft Azure

En la segunda etapa de la presente práctica, se realizó el despliegue de los sistemas en la plataforma Microsoft Azure, comenzando con la práctica correspondiente a los servicios web. Para ello, se utilizó la herramienta local Docker, con la cual se construyó una imagen del contenedor a partir del mismo Dockerfile previamente empleado en el despliegue en Google Cloud Platform (GCP). Una vez verificada la construcción local de la imagen, esta fue subida al repositorio Docker Hub, permitiendo su disponibilidad para el entorno en la nube.

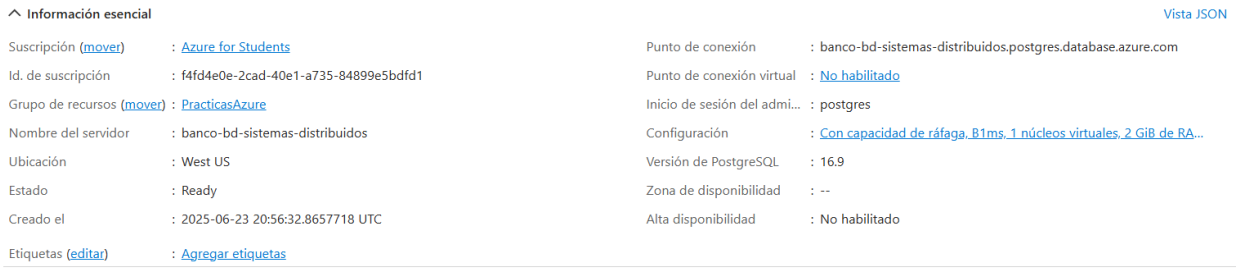


Posteriormente, se procedió al despliegue en Azure App Service, configurando el servicio para que apuntara directamente a la imagen almacenada en Docker Hub. Este enfoque facilitó el uso de contenedores personalizados dentro de Azure, lo que permitió mantener la coherencia con el entorno desplegado en GCP.

Una vez que el backend estuvo en línea, se actualizaron las rutas correspondientes en el frontend, el cual fue desplegado como un sitio web estático mediante el servicio Azure Static Web Apps. Debido a que se reutilizó el mismo contenedor previamente probado en GCP, no fue necesario realizar pruebas funcionales adicionales, ya que se asumía la integridad del entorno de ejecución.

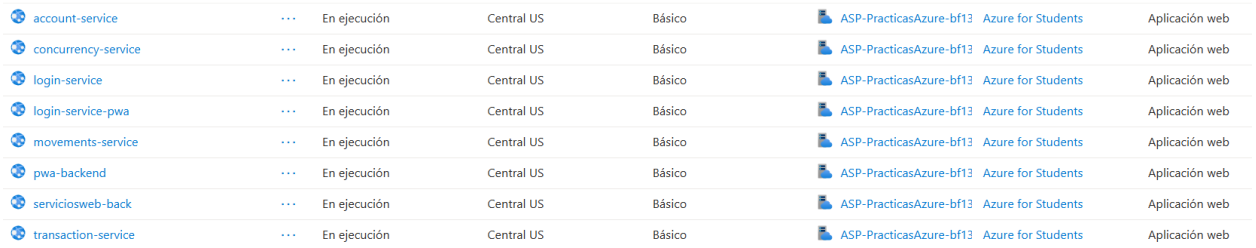


Continuando con la infraestructura base, se desplegó la base de datos PostgreSQL dentro del entorno de Azure. Para ello, se creó el recurso correspondiente y, mediante el acceso a consola (Azure Cloud Shell), se ejecutaron los comandos para crear la base de datos e inicializar las tablas con la estructura y datos necesarios, replicando el procedimiento realizado en GCP.

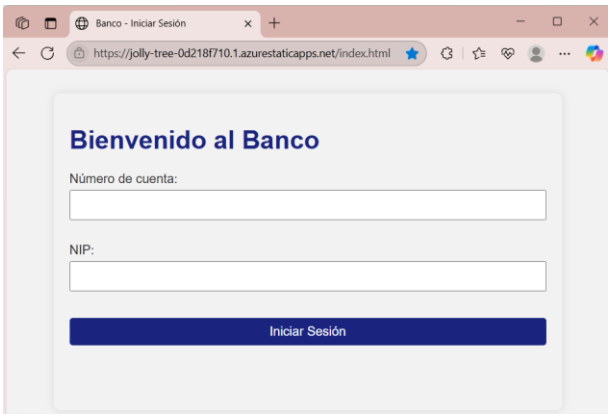


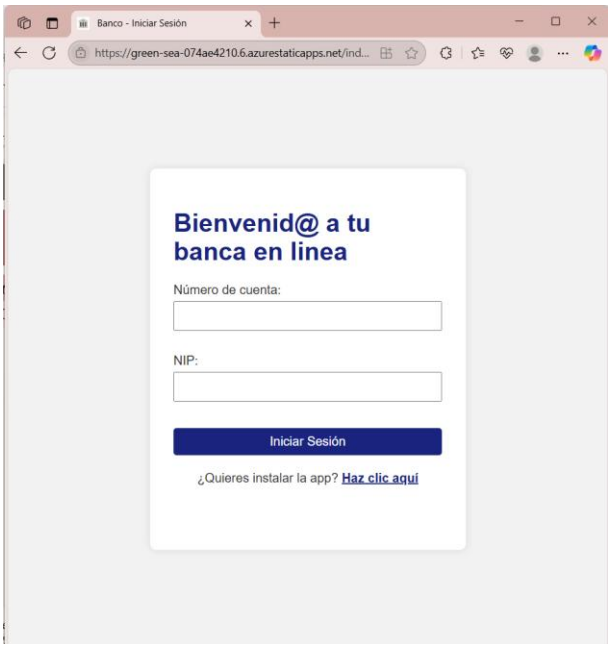
Una vez lista la base de datos, se procedió con el despliegue de los microservicios restantes, correspondientes tanto a la práctica de Microservicios como a la de PWA. Estos microservicios, previamente divididos en proyectos independientes, fueron empaquetados en sus respectivos contenedores utilizando Docker y almacenados en Docker Hub.

Posteriormente, fueron desplegados de forma individual en Azure App Service, configurando cada instancia con la imagen específica del microservicio correspondiente.



Con todos los microservicios desplegados y conectados a la base de datos en Azure, se actualizaron las URLs de consumo en las interfaces gráficas de las prácticas de Microservicios y PWA. Finalmente, ambas interfaces fueron desplegadas nuevamente como sitios web estáticos en Azure Static Web Apps, asegurando así la funcionalidad completa del sistema distribuido en la nube de Azure.



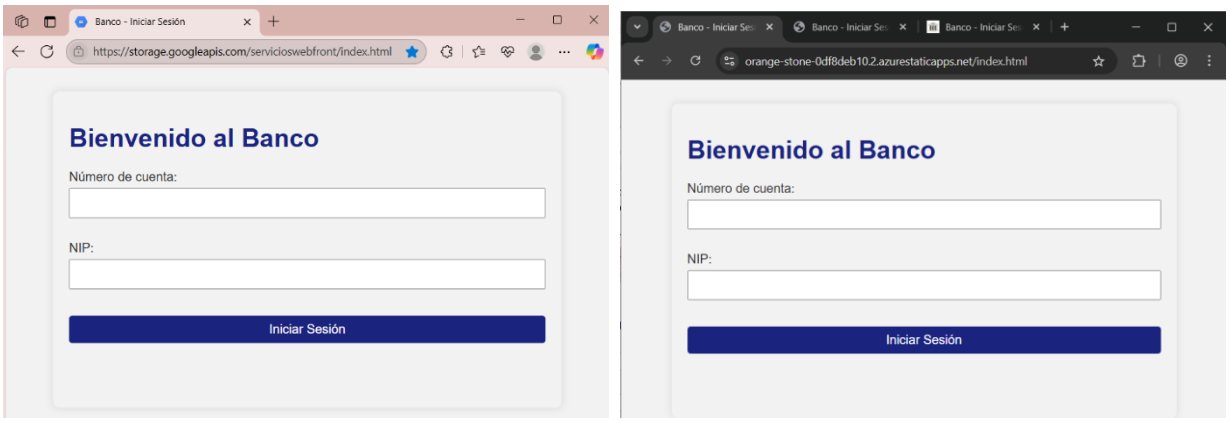


Este despliegue permitió validar la portabilidad y escalabilidad de la arquitectura diseñada, al comprobar que el mismo conjunto de microservicios, imágenes Docker y frontends podían ser ejecutados sin modificaciones significativas tanto en Google Cloud Platform como en Microsoft Azure.

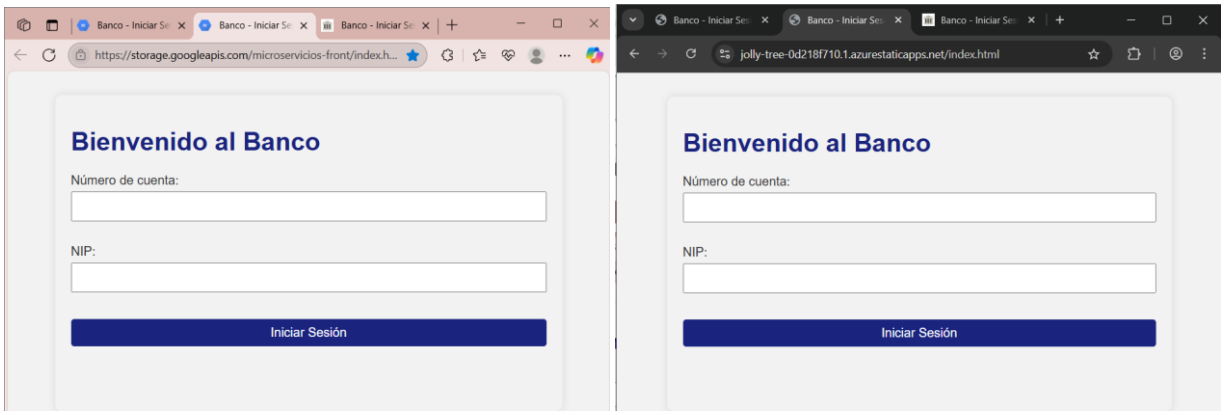
RESULTADOS

En esta sección se presentan los resultados obtenidos tras el despliegue y validación de las tres soluciones desarrolladas: servicios web, microservicios y aplicación web progresiva (PWA).

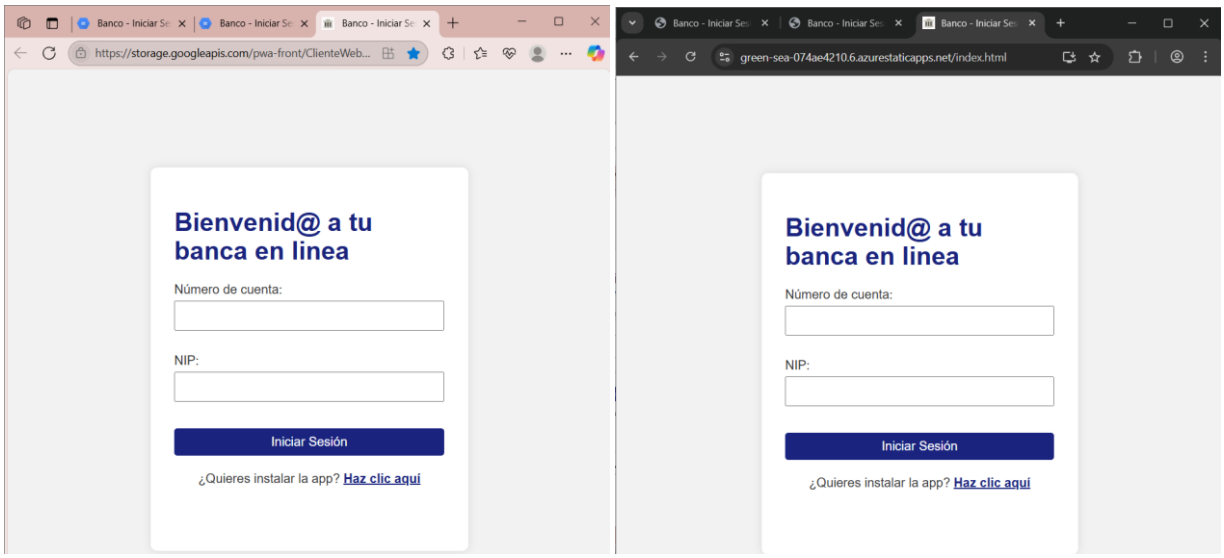
Al ingresar a los sitios web, se nos recibe con la pantalla de inicio del sistema bancario



Servicios Web en GCP y Azure

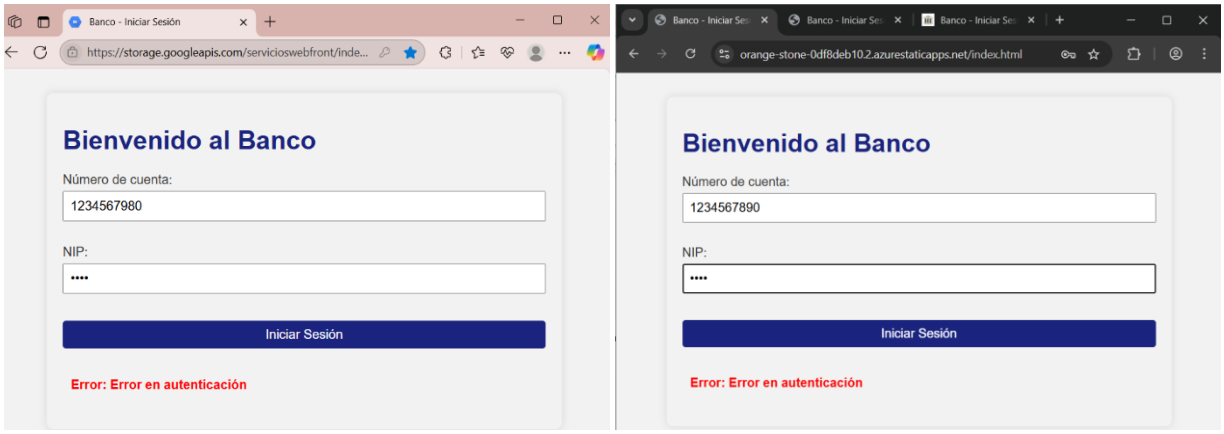


Microservicios en GCP y Azure

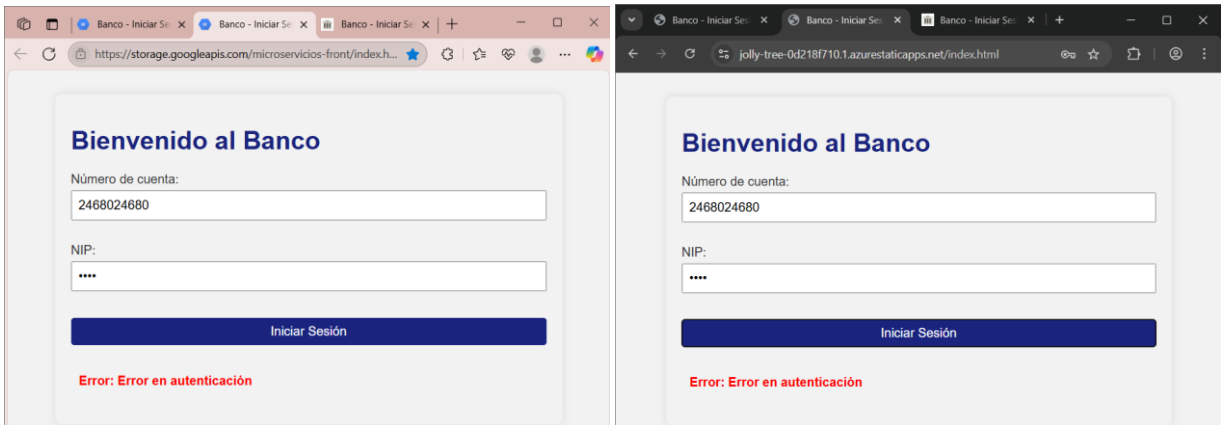


PWA en GCP y Azure

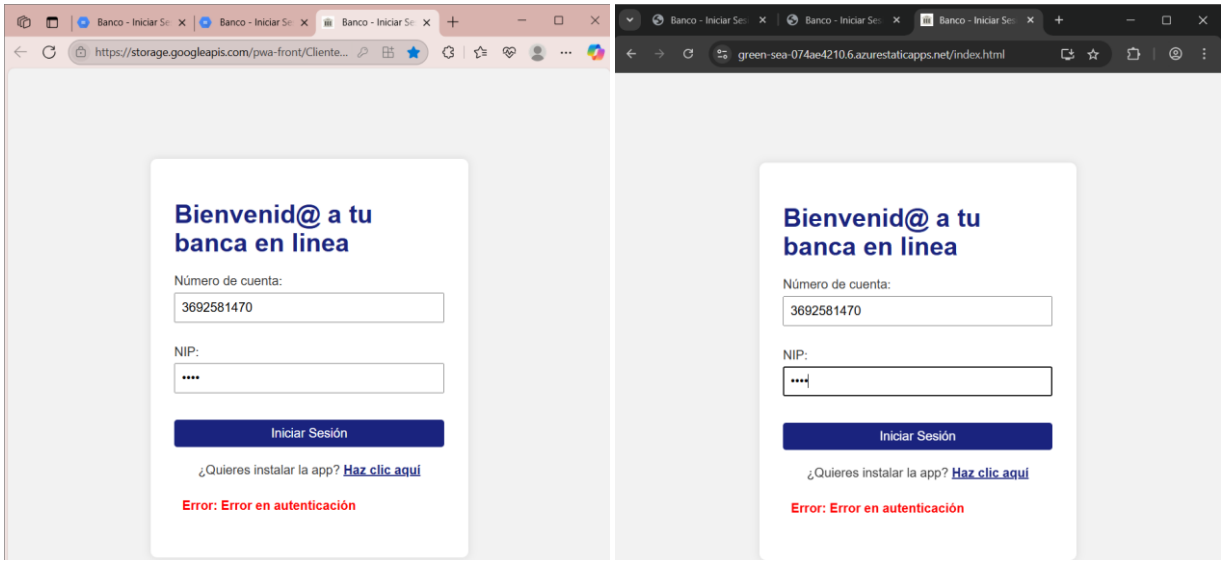
Posteriormente ingresamos datos incorrectos de ingreso, donde el sistema valido y marco el error de autenticación.



Servicios Web en GCP y Azure

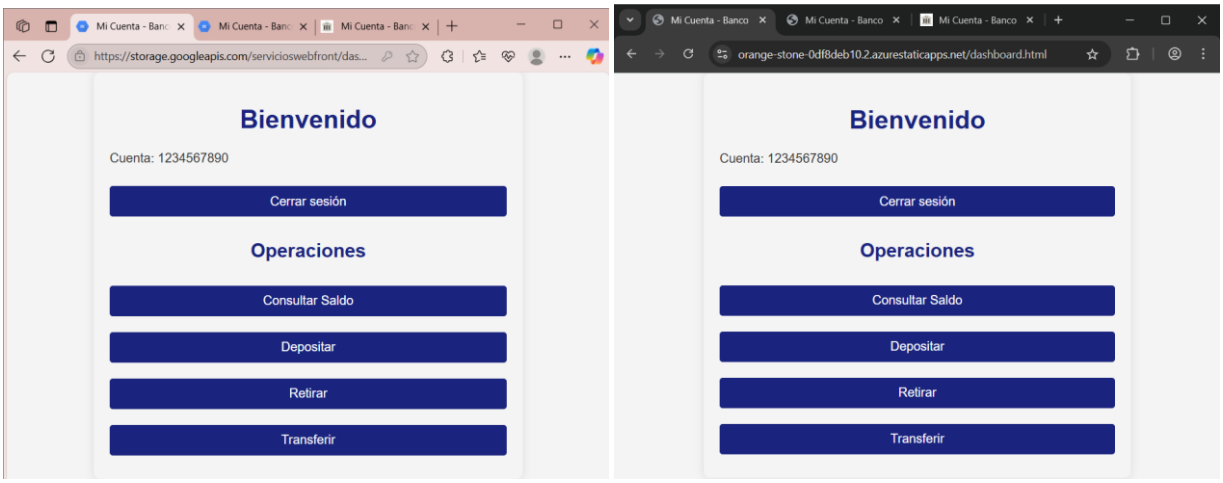


Microservicios en GCP y Azure

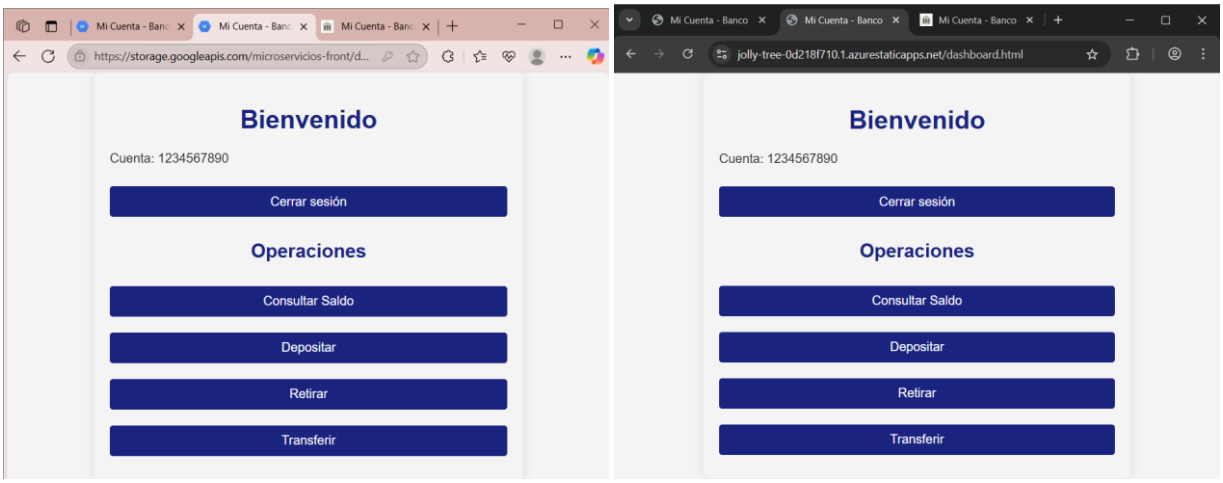


PWA en GCP y Azure

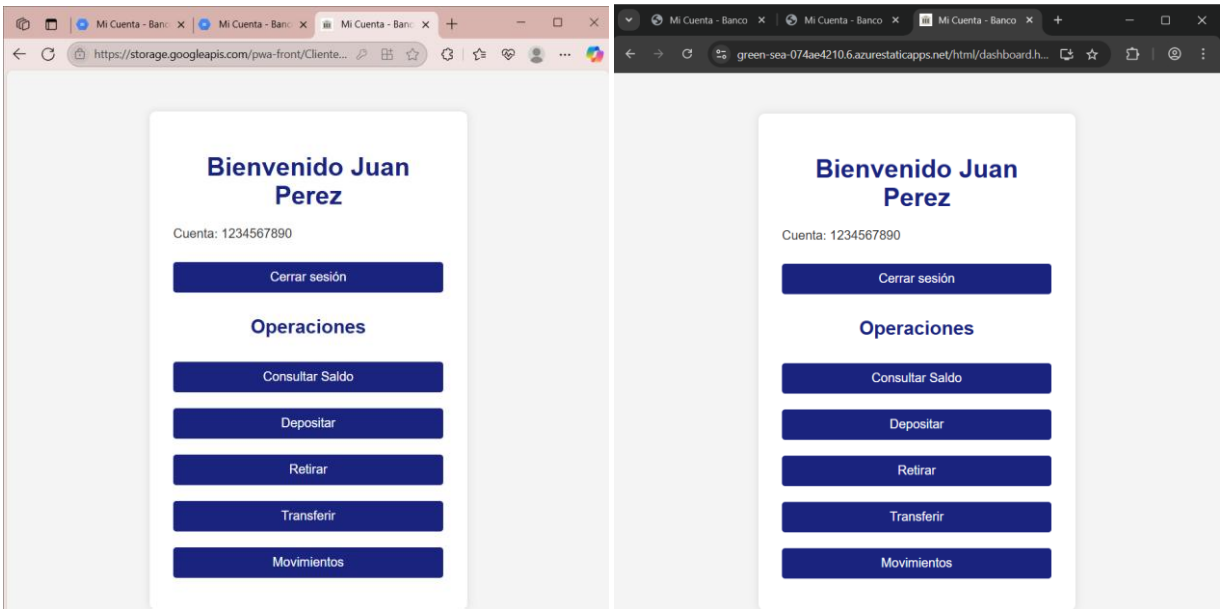
Enseguida, ya dentro del sistema, tenemos el siguiente dashboard, donde podemos generar diversas operaciones, como depositar dinero a la cuenta, retirar dinero de la cuenta y enviar una trasferencia a otra cuenta.



Servicios Web en GCP y Azure

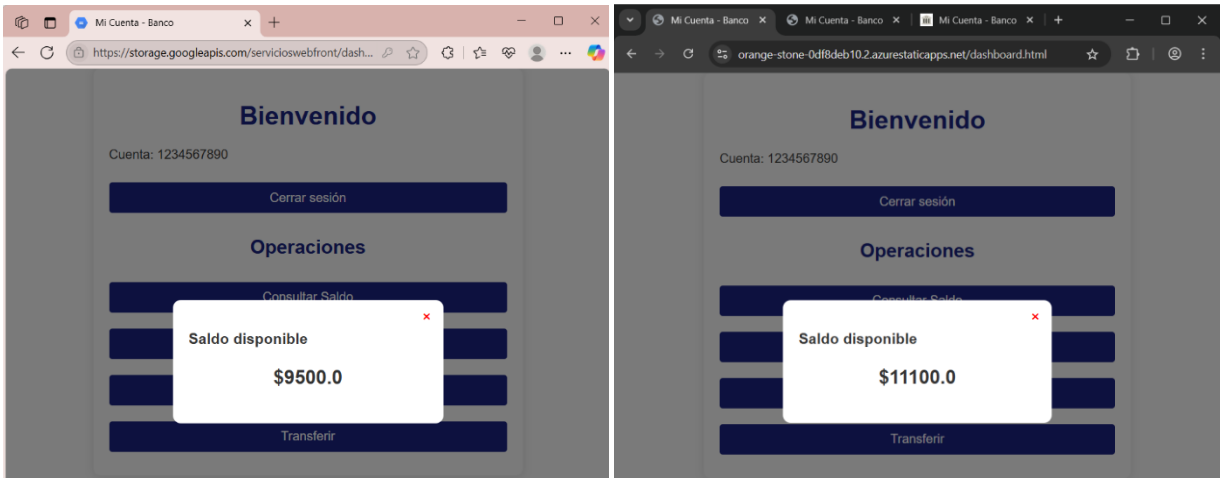


Microservicios en GCP y Azure

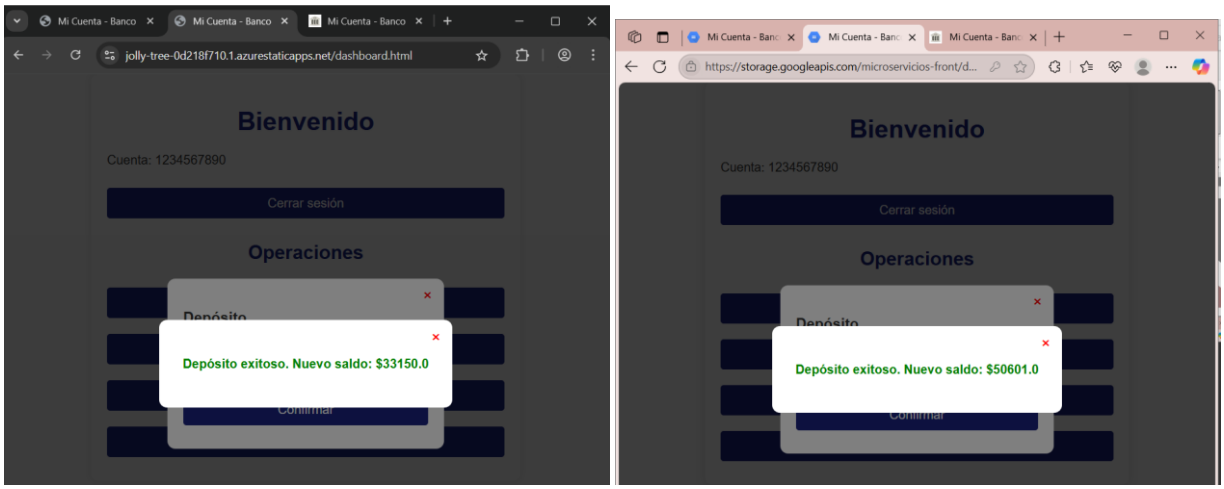


PWA en GCP y Azure

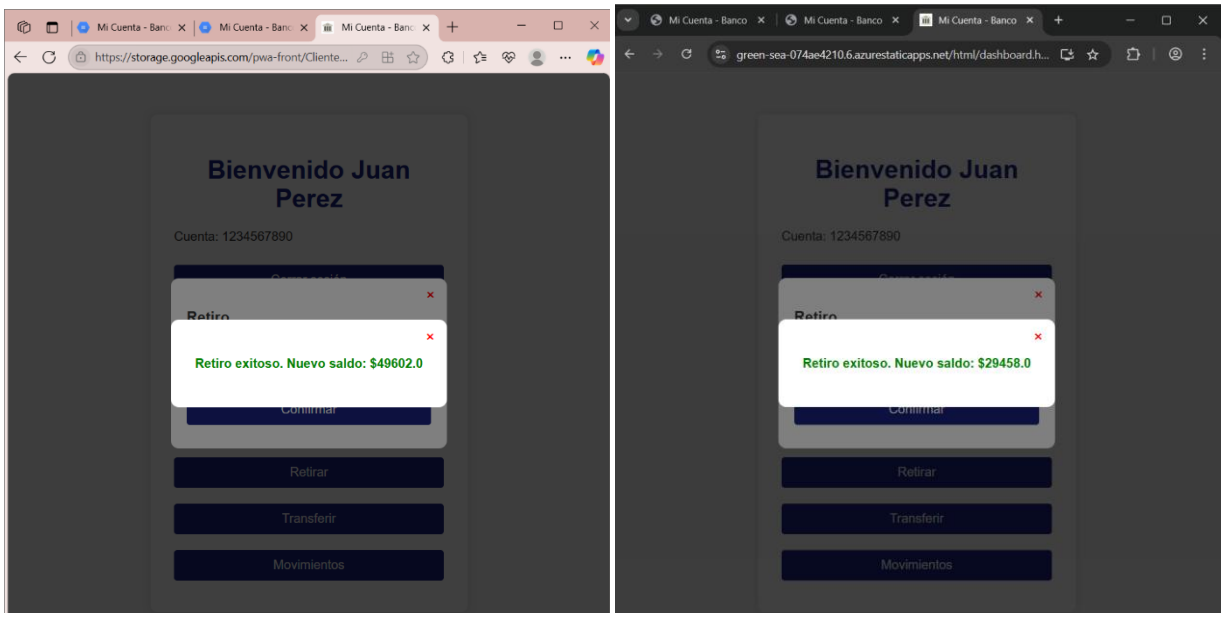
El sistema nos permite realizar diversas operaciones dentro del sistema como lo que es consultar el saldo de la cuenta, realizar un deposito a la cuenta, retirar una determinada cantidad de dinero, realizar una trasferencia a otra cuenta y revisar los movimientos realizados, aunque esta última función es única de PWA.



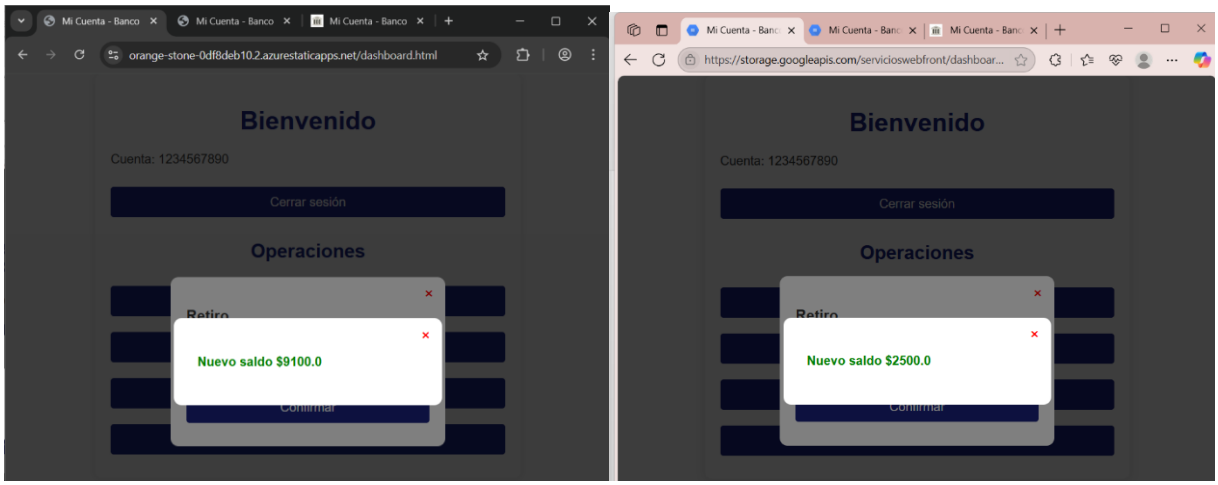
Consulta de saldo en sistema Servicios Web en GCP y Azure



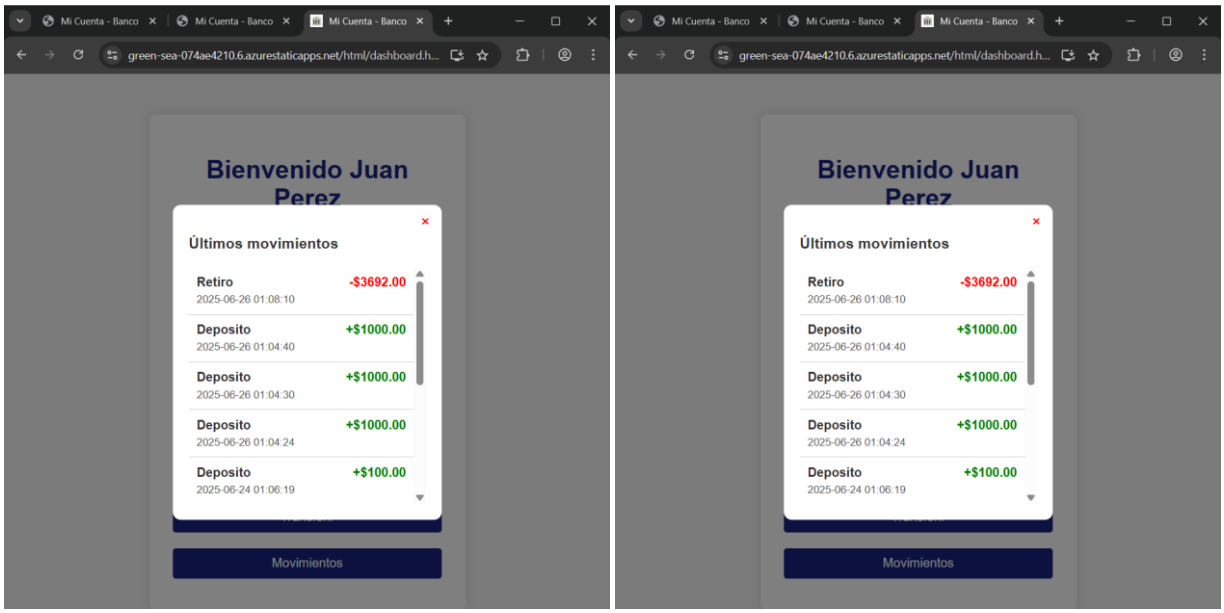
Depositó de dinero en microservicios en Azure y GCP



Retiro de efectivo en PWA en GCP y Azure



Transferencia de dinero en servicios web en Azure y GCP



Revisión de movimientos en PWA en GCP y Azure

CONCLUSIONES

El despliegue de los sistemas desarrollados, servicios web, microservicios y PWA, en plataformas de nube como Google Cloud Platform (GCP) y Microsoft Azure permitió comprender a profundidad los retos y beneficios de migrar hacia arquitecturas modernas basadas en contenedores. Uno de los principales desafíos fue la necesidad de reorganizar la práctica original en microservicios independientes. Esta reestructuración implicó separar la lógica en distintos proyectos, generar Dockerfiles específicos por cada componente y garantizar que todos pudieran conectarse de forma correcta a una base de datos centralizada, alojada en PostgreSQL. Aunque esto representó un esfuerzo considerable, fue clave para lograr una arquitectura desacoplada, escalable y fácilmente mantenible.

En el caso particular de Azure, se identificó una limitación importante: la imposibilidad de vincular directamente repositorios de GitHub para generar contenedores de forma automática, como sí se permitió en GCP mediante Cloud Build y Cloud Run. Esta restricción obligó a migrar a un enfoque apoyado en Docker Hub, donde se almacenaron las imágenes de los contenedores previamente generadas localmente. Posteriormente, estas imágenes fueron consumidas por Azure App Service para su despliegue. Aunque el proceso fue más manual y fragmentado, permitió consolidar un flujo funcional de despliegue.

Además, la creación y despliegue de la base de datos PostgreSQL tanto en GCP como en Azure fue fundamental para validar la persistencia de los servicios y garantizar la consistencia de la información. La integración de todos los microservicios con esta base de datos fue esencial para pruebas de login, registro de movimientos y consultas de saldo, funciones clave en los sistemas desarrollados.

Otro punto relevante fue la construcción e integración de la PWA, que reutilizó microservicios anteriores e incorporó nuevos servicios específicos para autenticar al usuario y gestionar operaciones bancarias. Su despliegue exitoso en GCP como sitio web estático, conectado a servicios distribuidos, demostró la viabilidad de un ecosistema completo y moderno basado en tecnologías web.

Este proceso permitió no solo aplicar conocimientos técnicos en desarrollo, contenedorización y despliegue, sino también comprender las diferencias prácticas entre plataformas de nube. Se reforzaron habilidades en CI/CD, en gestión de contenedores con Docker y en resolución de problemas reales relacionados con infraestructura y configuración. En conjunto, la experiencia adquirida proporciona una base sólida para futuros desarrollos basados en microservicios, servicios en la nube y aplicaciones modernas multiplataforma.