

# Palace

Financial Markets Technology for Latam

## GCP Security Example App

Descripción de configuración

Junio 2025

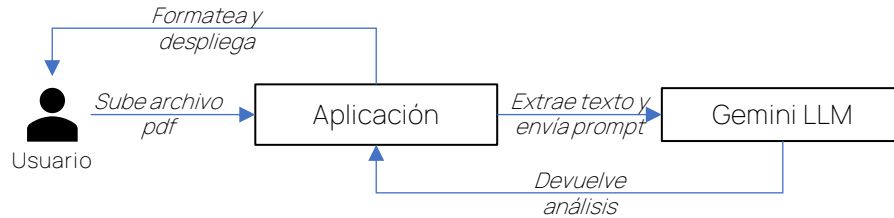


**Palace**

Financial Markets Technology for Latam

# Introducción: una aplicación simple, genera inquietudes de seguridad...

Proceso funcional conceptual:



Prompt: *Please tell me the main points of the document below, including the two main parties involved, in the following format:*

*Counterparty 1: (counterparty 1 name)*  
*Counterparty 2: (counterparty 2 name)*  
*Agreement Date: (agreement date)*  
*Agreement Type: (agreement type)*

*Do not provide any other information.*

Provoca unas preguntas de seguridad:



¿Cómo asegurar que solo usuarios válidos pueden usar la aplicación?



¿Cómo enviar y almacenar seguramente un archivo con información sensible?



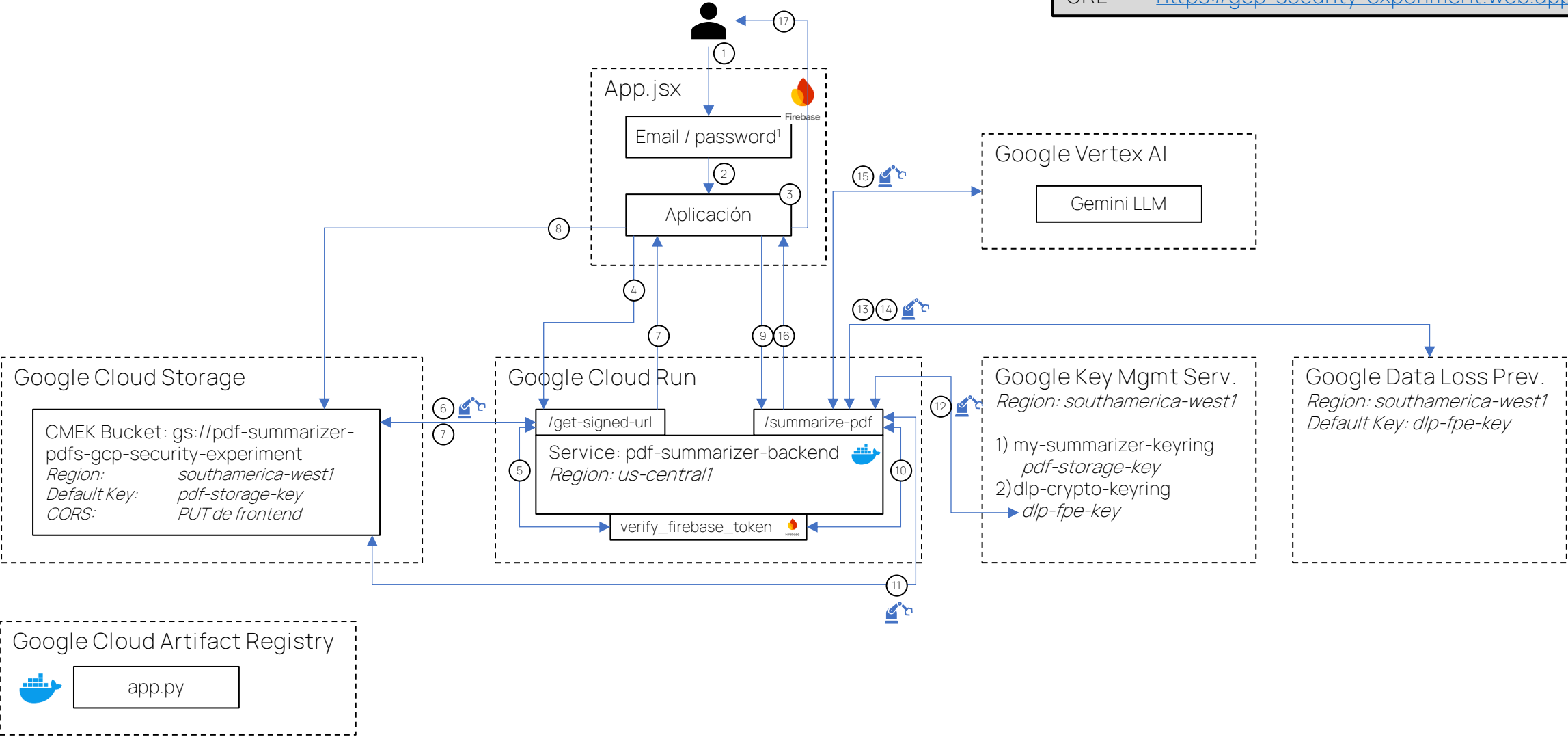
¿Cómo prohibir entrar directamente al backend e intervenir la aplicación?



¿Cómo evitar enviar información sensible al LLM?

# Resumen de arquitectura

Project: gcp-security-experiment  
Region: us-central1  
URL: <https://gcp-security-experiment.web.app/>



1 – App de ejemplo usa Email/Password. Producción usaría Enterprise SSO (SAML / OpenID Connect) mediante Firebase Identity Platform  
Service Account: pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com

# Puntos no implementados aún

A mencionar, pero discutir al final:

*Requiere configuración de una Organización en GCP:*

- 1) Google Virtual Private Cloud Service Controls (VPC-SC)
- 2) Limitar acceso a objetos en Bucket a Service Accounts
- 3) Security Command Center (SCC)

*No se hizo por priorizar MVP:*

- 4) API Gateway
- 5) Persistencia en Base de Datos (Firestore)
- 6) Setup de dashboards de Cloud Monitoring
- 7) Configuración de Cloud Audit Logs

*Difícil de hacer "standalone":*

- 8) SSO (SAML / OpenID Connect) con Firebase Identity Platform

*Otras dudas:*

- 9) Setup de región: data residency vs servicios disponibles
- 10) Dificultad con `-allow-unauthenticated` en deployment de backend en Cloud Run

# Descripción de pasos 1-9

- 1) Usuario va a URL del frontend: <https://gcp-security-experiment.web.app/>
- 2) Usuario hace login. Se valida contra Firebase (email/password). Se genera un JSON Web Token (JWT) indicando que usuario está autenticado.
- 3) Usuario selecciona un PDF a analizar.
- 4) Con JWT válido, el frontend llama a CLOUD\_RUN\_URL/get-signed-url, pasando JWT en el header junto con nombre del PDF y el tipo de archivo.
- 5) El primer paso del API de get-signed-url es verificar el JWT de Firebase con el decorator. En este proceso, toma el JWT y lo valida contra Firebase.
- 6) Una vez autenticado, se genera un nombre para el archivo a subir a Google Cloud Storage. Llama al API de Cloud Storage usando el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene permisos para operar en el Cloud Bucket, pasándole el nombre del archivo que acaba de generar.
- 7) El Service Account usa su propia key privada para generar un signed URL para el archivo en el Cloud Bucket y lo devuelve a la función get\_signed\_url. Este URL y el nombre del archivo se devuelven al Frontend que llamó al API de get-signed-url. El efecto de esto es darle al frontend un pase temporal (validado por el SA del backend) para hacer un PUT (no leer o modificar) de este archivo específico en el Bucket. No tenemos que exponer las credenciales de Google Cloud al frontend.
- 8) El frontend usa el signedURL para subir el PDF al Cloud Bucket. El CORS policy en el Cloud Bucket permite al frontend entrar a ejecutar este proceso.
- 9) Luego, el frontend llama al API CLOUD\_RUN\_URL/summarize-pdf, nuevamente pasando el JWT en el header y el nombre del PDF en el Cloud Bucket.

## Descripción de pasos 10-14

- 10) El primer paso del API de summarize-pdf es verificar el JWT de Firebase con el decorator. En este proceso, toma el JWT y lo valida contra Firebase.
- 11) Una vez autenticado, utilizando el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](https://pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene permisos para operar en el Cloud Bucket, le pasa el nombre del archivo que acaba de recibir, recibe el archivo, y extrae el texto del mismo.
- 12) Luego, el backend genera un Data Encryption Key (texto aleatorio) y lo encripta (wrapped) usando Key Management Services (KMS), específicamente con la key llamada "dlp-fpe-key". Esto se hace para asegurar que la DEK nunca se almacene o se exponga en texto abierto. Acceda al KMS usando el Service Account, ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](https://pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)), mediante los Application Default Credentials (ADCs).
- 13) Posteriormente, toma el texto del PDF y define tipos de información que espera encontrar en él, siendo tipos estándar o custom (ej. Counterparty Name). Define cómo "de-identificar" estos datos: Para "Counterparty Name", usa la DEK (key de encriptación de datos) previamente "wrappeadas" y el nombre de la key maestra de KMS "dlp-fpe-key". DLP internamente "des-wrapea" la DEK usando la key maestra de KMS, y luego utiliza la DEK (ahora en texto abierto dentro del entorno seguro de DLP) para realizar la encriptación de formato preservado (FPE) de la contraparte. Para los otros tipos de dato, define el método de redacción como método de de-identificarlos (ej. EMAIL\_ADDRESS).
- 14) Manda toda esta información a Google Data Loss Prevention (DLP) que devuelve el texto del PDF ya pseudonimizado (para las contrapartes) y redactado (para otros datos sensibles). Para permitir la re-identificación de las contrapartes en la respuesta final, el Backend previamente ha calculado, de forma determinística con DLP, el pseudónimo que corresponde a cada contraparte conocida. Esta correspondencia se guarda en una lista (mapa) en memoria del Backend.

## Descripción de pasos 15-17

- 15) El backend manda el prompt, que incluye el texto anonimizado, a Gemini en Vertex AI. Utiliza el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene acceso a usar el API de Gemini, mediante los Application Default Credentials (ADCs).
- 16) Gemini devuelve una respuesta al prompt, pero no sabe los nombres de las contrapartes y otros datos que fueron anonimizados. Aquí, el backend utiliza la lista de contrapartes y sus pseudónimos (previamente generada con DLP) para reemplazar los pseudónimos en la respuesta de Gemini con los nombres originales de las contrapartes. Se devuelve al frontend la respuesta original del LLM (aún con pseudónimos) y la respuesta final con los nombres reales re-identificados..
- 17) El frontend muestra las diferentes alternativas de texto al usuario.



# Demonstración de Seguridad

Acceder a URLs de backend directamente:

- 1) `curl -X POST -H "Content-Type: application/json" -d '{"objectName": "test"}'` <https://pdf-summarizer-backend-459042076639.us-central1.run.app/get-signed-url>
- 2) `curl -X POST -H "Content-Type: application/json" -d '{"objectName": "test"}'` <https://pdf-summarizer-backend-459042076639.us-central1.run.app/summarize-pdf>

Acceder a PDFs en Cloud Storage:

- 1) Cloud Storage permissions: <https://console.cloud.google.com/storage/browser/pdf-summarizer-pdfs-gcp-security-experiment;tab=permissions?forceOnBucketsSortingFiltering=true&inv=1&inv=Ab0XNA&project=gcp-security-experiment&prefix=&forceOnObjectsSortingFiltering=false>
- 2) Acceso directo a archivo por URL: [https://storage.googleapis.com/pdf-summarizer-pdfs-gcp-security-experiment/<file\\_name>](https://storage.googleapis.com/pdf-summarizer-pdfs-gcp-security-experiment/<file_name>)



Anexos



**Palace**

Financial Markets Technology for Latam

# Resumen

# Resumen

## Google Service Accounts

SA: pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com

Roles: *aiplatform.user*

*(use Vertex AI Gemini API)*

*artifactregistry.reader*

*(allow Cloud Run to pull the Docker image)*

*firebase.viewer & iam.serviceAccountTokenCreator*

*(to verify Firebase ID token)*

*storage.objectCreator & storage.objectViewer*

*(to create (upload) and read (download) objects from the bucket)*

*iam.serviceAccountTokenCreator*

*(to sign V4 URLs using its own credentials)*

*cloudkms.viewer & cloudkms.cryptoKeyEncrypterDecrypter*

*(to use the KMS key for Format Preserving Encryption (FPE) on DLP)*

SA: service-`<PROJECT_NUMBER>`@gs-project-accounts.iam.gserviceaccount.com

*kms authorize pdf-storage-key*

## Google Key Management Service (KMS)

Key Ring: my-summarizer-keyring (southamerica-west1):

*pdf-storage-key*

*dlp-crypto-keyring* (southamerica-west1)

*dlp-fpe-key*

## Google Virtual Private Cloud Service Controls (VPC-SC)

No se pudo hacer porque es un servicio a nivel de Organización, costo-beneficio no da para este ejemplo. Pero se quiere hacer en producción (revisar con Max).

API Gateway???

Cloud Logging – está por defecto

Cloud Monitoring – está por defecto, config de dashboards?

Cloud Audit Logs – configurar, algo hice mal

Security Command Center (SCC) – conceptual, req. Org.



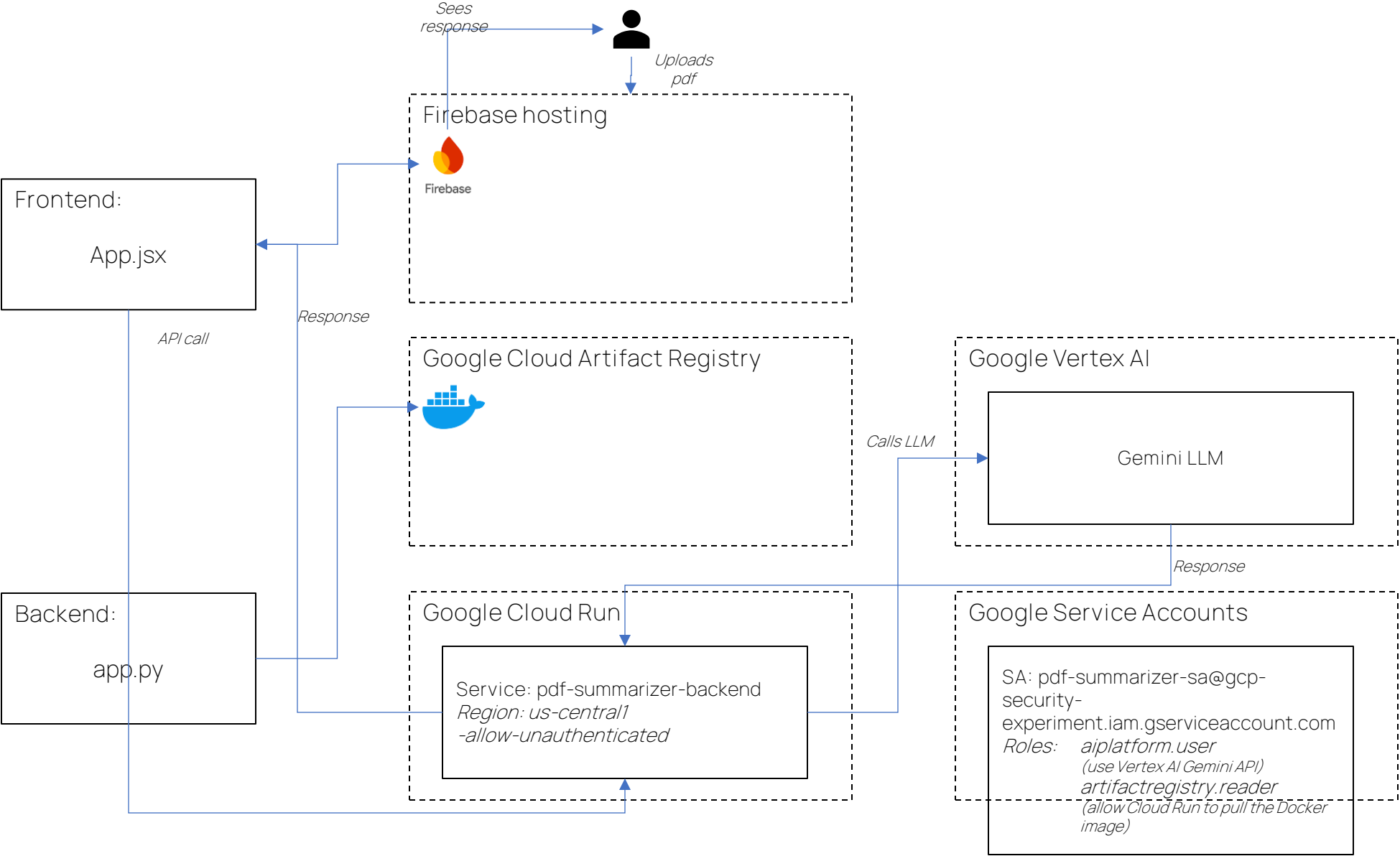
**Palace**

Financial Markets Technology for Latam

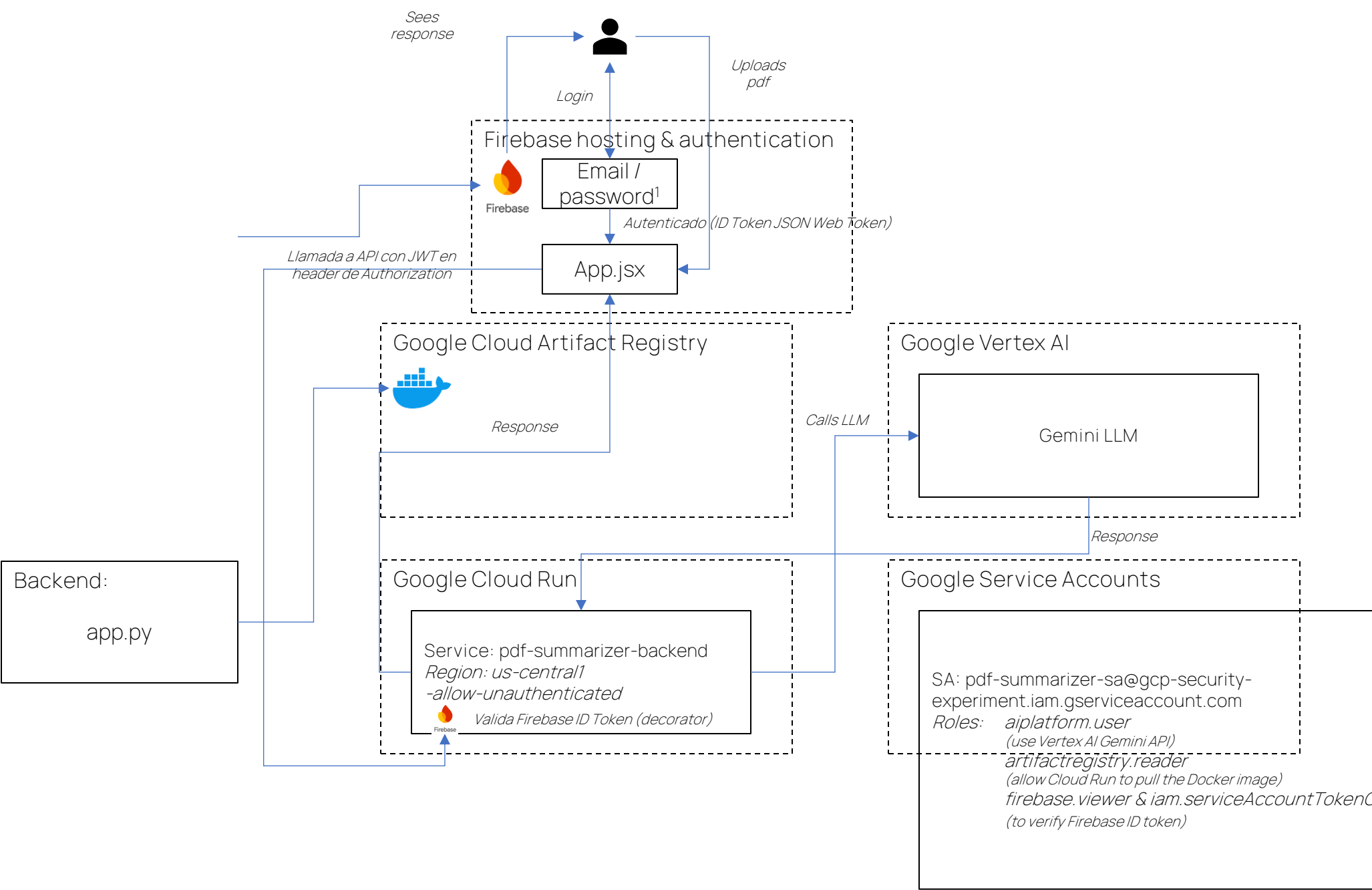
# Descripción de pasos

- 1) Usuario va a URL: <https://gcp-security-experiment.web.app/>
- 2) Usuario hace login. Validado contra Firebase (email/password). Se genera JSON Web Token (JWT) indicando que usuario está autenticado.
- 3) Usuario selecciona un PDF a analizar.
- 4) Con JWT válido, el frontend llama a CLOUD\_RUN\_URL/get-signed-url, pasando JWT en header junto con nombre del PDF y el tipo de archivo.
- 5) El primer paso del API de get-signed-url es verificar el JWT de Firebase con el decorator (the bouncer). Aquí, toma el JWT y lo valida contra Firebase.
- 6) Una vez autenticado, genera un nombre para el archivo a subir a Google Cloud Storage. Llama al API de Cloud Storage usando el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene permisos para operar en el Cloud Bucket, pasándole el nombre del archivo que acaba de generar.
- 7) El Service Account usa su propia key privada para generar un signed URL para el archivo en el Cloud Bucket y lo devuelve a la función get\_signed\_url. Este URL y el nombre del archivo se devuelven al Frontend que llamó al API de get-signed-url. El efecto de esto es darle al frontend un pase temporal (validado por el SA del backend) para hacer un PUT (no leer o modificar) de este archivo específico en el Bucket. No tenemos que exponer las credenciales de Google Cloud al frontend.
- 8) El Frontend usa el signedURL para subir el PDF al Cloud Bucket. El CORS policy en el Cloud Bucket permite al frontend entrar a ejecutar este proceso.
- 9) Luego, el frontend llama al API CLOUD\_RUN\_URL/summarize-pdf, nuevamente pasando el JWT en el header y el nombre del PDF en el Cloud Bucket.
- 10) El primer paso del API de summarize-pdf es verificar el JWT de Firebase con el decorator (the bouncer). Aquí, toma el JWT y lo valida contra Firebase.
- 11) Una vez autenticado, utilizando el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene permisos para operar en el Cloud Bucket, le pasa el nombre del archivo que acaba de recibir, recibe el archivo, y extrae el texto del mismo.
- 12) Luego, el backend genera un Data Encryption Key (texto aleatorio) y lo encripta (wrapped) usando Key Management Services (KMS), específicamente con la key llamada "dlp-fpe-key". Acceda al KMS usando el Service Account, ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)), mediante los Application Default Credentials (ADCs).
- 13) Posteriormente, toma el texto del PDF y define tipos de información que espera encontrar en él, siendo tipos estándar o custom (ej. Counterparty Name). Define cómo "de-identificar" estos datos: Para "Counterparty Name", usa la DEK (key de encriptación de datos) previamente "wrappeadas" y el nombre de la key maestra de KMS "dlp-fpe-key". DLP internamente "des-wrapea" la DEK usando la key maestra de KMS, y luego utiliza la DEK (ahora en texto plano dentro del entorno seguro de DLP) para realizar la encriptación de formato preservado (FPE) de la contraparte. Para los otros tipos de dato, define el método de redacción como método de de-identificarlos (ej. EMAIL\_ADDRESS).
- 14) Manda toda esta información a Google Data Loss Prevention (DLP) que devuelve el texto del PDF ya pseudonimizado (para las contrapartes) y redactado (para otros datos sensibles). Para permitir la re-identificación de las contrapartes en la respuesta final, el Backend previamente ha calculado, de forma determinística con DLP, el pseudónimo que corresponde a cada contraparte conocida. Esta correspondencia se guarda en una lista (mapa) en memoria del Backend.
- 15) Mandamos el prompt, que incluye el texto anonimizado, a Gemini en Vertex AI. Utiliza el Service Account ([pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com](mailto:pdf-summarizer-sa@gcp-security-experiment.iam.gserviceaccount.com)) que tiene acceso a usar el API de Gemini, mediante los Application Default Credentials (ADCs).
- 16) Gemini nos devuelve una respuesta al prompt, pero no sabe los nombres de las contrapartes y otros datos que fueron anonimizados. Aquí, el Backend utiliza la lista de contrapartes y sus pseudónimos (previamente generada con DLP) para reemplazar los pseudónimos en la respuesta de Gemini con los nombres originales de las contrapartes. Devolvemos al frontend la respuesta original del LLM (aún con pseudónimos) y la respuesta final con los nombres reales re-identificados..
- 17) El frontend muestra las diferentes alternativas de texto al usuario.

# Resumen

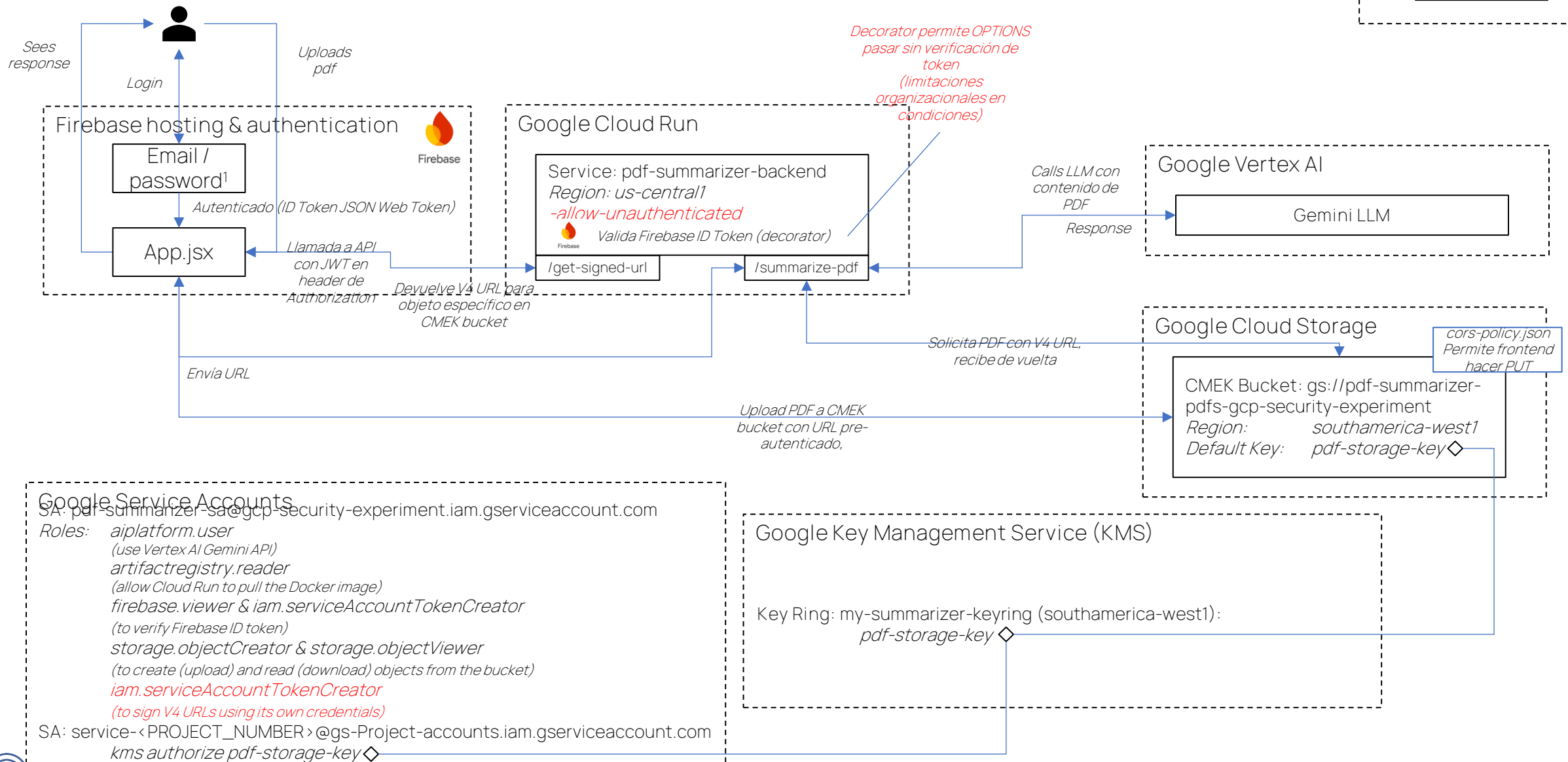
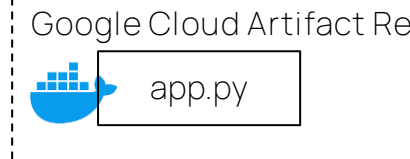


# Resumen



# Resumen

Project: gcp-security-experiment






# Resumen

Project: gcp-security-experiment  
Region: us-central1

Google Cloud Artifact Re

 app.py

