Elegí una arquitectura ‘serverless’ (Función-como-Servicio o FaaS) porque entre sus ventajas encontré que provee capacidad de cómputo para ejecutar nuestro código de aplicación sin necesidad de gestionar servidores, siendo de esta forma el proveedor de ‘cloud’ el que ejecuta el código (llamado **función**) sólo cuando se necesita, y escala automáticamente para satisfacer la demanda. Presenta también como ventajas a considerar una administración totalmente automatizada, tolerancia a errores integrada, escalado automático, nos permite también controlar el rendimiento en detalle (incrementando la simultaneidad cuando sea necesario en periodos de alta demanda y disminuir cuando la misma disminuya), puede también hacer lectura y escrituras de grandes volúmenes de datos, la seguridad viene integrada también y se paga en función de los recursos que se consumen y finalmente todo el código es ejecutado en una infraestructura de alta disponibilidad.

También nos proporciona un nivel elevado de desacoplamiento entre los diferentes servicios, favoreciendo el desarrollo de arquitecturas basadas en micro servicios. Esto facilita mucho el ciclo de vida y los despliegues continuos.

Por otro lado, el factor a considerar seria el costo del escalamiento de la arquitectura en comparación con utilizar, por ejemplo EC2, también un servicio de Amazon, pero basándome en la información de la web, para determinar qué modelo de arquitectura es mejor que otro habría que tener mayor conocimiento del dominio, ya que una variedad de factores específicos de cada servicio puede determinar cuál es la mejor elección.

Arquitectura:

Amazon API Gateway: Sirve para publicar funciones en Internet. Es la puerta de entrada para acceder a la mayoría de los servicios en la nube, siendo este un recurso en la nube sin servidor. Ofrece funciones como: el almacenamiento en caché, seguridad, aceleración de las solicitudes, por nombrar las mas elementales.

Funciones lambda: Cada función incluye código y cierta información de configuración asociada, incluidos el nombre de la función y los requisitos en materia de recursos. Las funciones de Lambda "no tienen estado" y no tienen ninguna afinidad con la infraestructura subyacente, por lo que Lambda puede lanzar rápidamente tantas copias de la función como resulten necesarias para ajustar la escala al índice de eventos entrantes.

S3 bucket: Se utiliza como almacenamiento de objetos, donde podemos guardar imágenes o recibir datos de texto para ser procesados, como un archivo CSV para ser importado a las bases de datos.

Amazon Aurora: Es una base de datos relacional altamente escalable, replica sus datos sin afectar el rendimiento de la base de datos, permite lecturas locales rápidas con baja latencia en cada región y proporciona recuperación de desastres de interrupciones en toda la región. Es la mejor opción para trabajar junto con las funciones Lambda altamente escalables. Se paga por E/S de escritura replicadas entre la región principal y cada una de las regiones secundarias, pero es relativamente bajo.

Amazon DynamoDB: Base de datos de noSQL sin servidor altamente escalable, donde podemos acceder a datos con latencia muy baja. Es una base de datos duradera de varias regiones y con varios maestros, completamente administrada, que cuenta con copia de seguridad, restauración y seguridad integradas, y almacenamiento de caché en memoria para aplicaciones a escala de Internet. DynamoDB puede gestionar más

de 10 billones de solicitudes por día y puede admitir picos de más de 20 millones de solicitudes por segundo.

Modelo de arquitectura propuesto: