ELITA PROJECTS

Análisis y Visión General

- **Colaboradores:** Eduardo de Andrés
- **Área general de desarrollo:** Integración sistemas de voz con IA
- **Descripción del documento:**
    - Visión general y anotaciones para dar una visión general del proyecto y mejorar su comprensión para el jurado de Hackaton for Good Iberia 2023.

Buenas prácticas de ElitaProjects:

1. **Define los objetivos y el público objetivo**: Antes de comenzar a escribir, asegúrate de tener claro cuál es el objetivo de la documentación y quiénes serán los lectores.
2. **Organiza la información**: Estructura tu documento de manera clara y coherente, utilizando títulos y subtítulos que guíen al lector por el contenido.
3. **Sé claro y conciso**: Utiliza un lenguaje sencillo y directo que sea fácil de entender. Evita jergas y tecnicismos innecesarios, y no te extiendas en detalles que no aporten valor al lector.
4. **Revisa y corrige**: Una vez que hayas terminado de escribir, revisa cuidadosamente tu documento para detectar errores de ortografía, gramática o formato. Pide a alguien más que lo revise también, ya que es posible que se te hayan pasado algunos errores.
5. **Actualiza regularmente**: Mantén tu documentación actualizada y relevante, especialmente si se trata de información técnica o de procedimientos que cambian con el tiempo.

Descripción del proyecto

Descripción general

Desde el grupo de ElitaProjects, nos motiva la idea de poder aplicar nuestros conocimientos tecnológicos, para desarrollar una nueva herramienta que permita mejorar la interacción entre los operarios y los atendidos por Cruz Roja. Basándonos en esta idea, queremos reducir la frialdad de las conversaciones mediante un software completamente automatizado y potenciado con inteligencia artificial para que la distracción de escribir textos y realizar análisis sea una tarea del pasado.

Para este proyecto no nos hemos enfocado únicamente en poder recolectar la información de una conversación y convertirla en texto. Nuestro objetivo va mucho más allá. Queremos poder captar las emociones y necesidades que nos muestran las personas a través de las conversaciones. Todo esto, sin olvidarnos de que el software tiene que ser accesible, escalable, fácil de usar y sobre todo que sea rápido, visualmente agradable y al menor coste posible.

La mejor combinación que todo lo cubre

Los dos componentes que forman la arquitectura principal para proporcionar la mejor experiencia de usuario desde todo tipo de dispositivo con conexión a internet, son Django (API) y React (Aplicativo Web).

Estas dos tecnologías poderosas, que combinadas, proporcionan una plataforma sólida para el desarrollo de aplicaciones web. React es una biblioteca de JavaScript para la creación de interfaces de usuario interactivas, mientras que Django es un framework de desarrollo web en Python. Juntos, proporcionan una combinación única de flexibilidad, escalabilidad y facilidad de uso para el desarrollo de aplicaciones web.

Una de las principales ventajas de combinar React y Django es la capacidad de crear aplicaciones altamente personalizables y escalables. Django proporciona una estructura sólida para la creación de aplicaciones web, mientras que React permite a los desarrolladores crear interfaces de usuario altamente personalizables. Esta combinación permite a los desarrolladores crear aplicaciones que se adapten a las necesidades específicas de los usuarios y que sean capaces de crecer y evolucionar a medida que las necesidades cambien.

Otra ventaja clave de esta combinación es la facilidad de uso. Tanto React como Django son tecnologías de código abierto con una amplia comunidad de desarrolladores que proporcionan soporte y documentación detallada. Esto significa que los desarrolladores pueden encontrar fácilmente recursos y soluciones para cualquier problema que puedan enfrentar durante el proceso de desarrollo.

Además, la combinación de React y Django proporciona una plataforma robusta y segura para el desarrollo de aplicaciones web. Django proporciona una serie de características de seguridad integradas, lo que garantiza que las aplicaciones sean seguras y confiables. React, por su parte, proporciona una estructura sólida para la creación de interfaces de usuario seguras y receptivas.

Diseñado para escalar y adaptarse a cualquier desafío, nuestro proyecto se basa en la arquitectura de React y Django, encapsulados en contenedores Docker para una eficiencia y portabilidad inigualables. No importa cuán grandes sean tus datos o cuán complejos sean tus procesos, nuestro sistema puede escalar fácilmente para satisfacer tus necesidades.

Con esta arquitectura, se mejora el uso en paralelo de una gran cantidad de usuarios. Por un lado los usuarios que estén realizando visualizaciones de datos, rellenado de documentos o cualquier otro tipo de acción que tenga que ver con la parte de React, no van a afectar directamente al rendimiento de Django que es el encargado de procesar los textos, aplicar la inteligencia artificial y realizar las estadísticas de los datos que se van obteniendo.

Todos nuestros modelos de programación están orientados en divide y vencerás, lo que nos lleva a la división de todas las funcionalidades y arquitecturas en micro servicios lo que nos permite reducir los errores de código, la redundancia y unos flujos de ejecución limpios.

Muy bien, ¿Pero qué nos puede ofrecer algo así?

1. **Experiencia de usuario**: React utiliza el Virtual DOM, una representación ligera del DOM real. Este enfoque minimiza las costosas operaciones de actualización del DOM real, lo que resulta en una aplicación más rápida y suave. Además como es un modelo que utiliza componentes esto permite crear una gran cantidad de páginas con los mismos formatos y estilos reduciendo el coste de hora de los programadores. También tiene incorporada una protección contra ataques de cros-site scripting mediante el escape automático de contenidos. Pero lo más importante es toda la cantidad de librerías y el ecosistema que tiene montado alrededor, lo que nos permite crear elementos de visualización estadísticos como todo tipo de gráficas, representación de mapas de datos y análisis estadísticos que se necesiten.
2. **Inteligencia Artificial**: mediante el uso de tecnologías de reinforcement learning y Transformers, nos encargamos de reducir el error de otro tipo de tecnologías y el error humano al mínimo. Desde el grupo de ElitaProjects siempre estamos al tanto de las nuevas tecnologías que salen al mercado y en este caso no iba a ser menos. Nuestras aplicaciones siempre se componen del uso de tecnologías de inteligencia artificial y en este caso de machine learning y la API de Chat GPT. Esto nos permite corregir las transcripciones y los errores de ortografía y gramaticales que se encuentran dentro de las transcripciones que realizan los atendidos por la Cruz roja. Posteriormente a ese texto corregido y con coherencia se le vuelve a aplicar inteligencia artificial para sacar por ejemplo el grado de necesidad que tiene una persona en los diferentes ámbitos marcados por la Cruz Roja. Pero esto es solo la punta del iceberg, la cantidad de utilidades y usos que se le pueden dar a la

inteligencia artificial para el análisis de texto actualmente es muy amplio, siempre y cuando se sepa utilizar y sacarle el máximo provecho para reducir al mínimo cualquier tipo de error producido por la IA.

3. **Estadísticas**: El análisis de texto, especialmente en el caso de transcripciones de atención al cliente, como las de la Cruz Roja, puede revelar información crítica y perspectivas valiosas. PySpark y NLTK (Natural Language Toolkit) son dos tecnologías potentes que se combinan a la perfección para desbloquear estas perspectivas. Aquí te explicamos por qué son la mejor opción para analizar y obtener estadísticas a partir de estos datos.
    3.1. **Escalabilidad con PySpark**: PySpark es la interfaz de Python para Spark, un motor de procesamiento de datos distribuido y en memoria, ideal para trabajar con grandes volúmenes de datos. PySpark permite procesar y analizar grandes conjuntos de datos de manera eficiente, algo que es fundamental cuando se trabaja con transcripciones de atención al cliente a gran escala. También ofrece una interfaz intuitiva y fácil de usar, lo que facilita el manejo de los datos y la implementación de algoritmos de procesamiento.
    3.2. **Procesamiento de lenguaje natural con NLTK:** NLTK es una biblioteca líder para el procesamiento de lenguaje natural (NLP) en Python. Permite realizar tareas como la tokenización, el etiquetado de partes del discurso, el análisis de sentimientos y la identificación de entidades, entre otras. Esto es fundamental para analizar transcripciones, ya que permite entender el contenido, el contexto y el sentimiento de las interacciones de los clientes.
    3.3. **Combinación perfecta para el análisis de texto:** La combinación de PySpark y NLTK permite analizar eficazmente grandes volúmenes de transcripciones. PySpark puede pre procesar y transformar los datos para prepararlos para el análisis de NLTK. Luego, NLTK puede profundizar en los detalles de las transcripciones, extrayendo información significativa y estadísticas a partir de los datos procesados.
    3.4. **Estadísticas y perspectivas valiosas:** Al utilizar PySpark y NLTK juntos, puedes extraer una gran cantidad de estadísticas y perspectivas valiosas de las transcripciones. Esto puede incluir la frecuencia de ciertas palabras o frases, el sentimiento general de las interacciones, temas comunes o problemas, y mucho más. Estos datos pueden ayudar a la Cruz Roja a mejorar su servicio, identificar áreas de preocupación y tomar decisiones más informadas.
4. **Integración y seguridad**: Django es una opción excelente para integrar tu API con todas las plataformas de la Cruz Roja por varias razones. Django es un framework de desarrollo web de alto nivel basado en Python que promueve el desarrollo rápido, el diseño limpio y pragmático, y la facilidad de uso, lo que lo convierte en una elección ideal para este propósito. Aquí te explicamos por qué:
    4.1. **Versatilidad y Flexibilidad:** Django es increíblemente flexible, lo que significa que se puede usar para construir casi cualquier tipo de aplicación web o API. Django viene con una arquitectura desacoplada, lo que facilita la integración con diferentes sistemas y tecnologías.
    4.2. **Seguridad:** Django ha sido diseñado con la seguridad en mente, proporcionando protección contra muchas vulnerabilidades comunes de la web, como Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF),

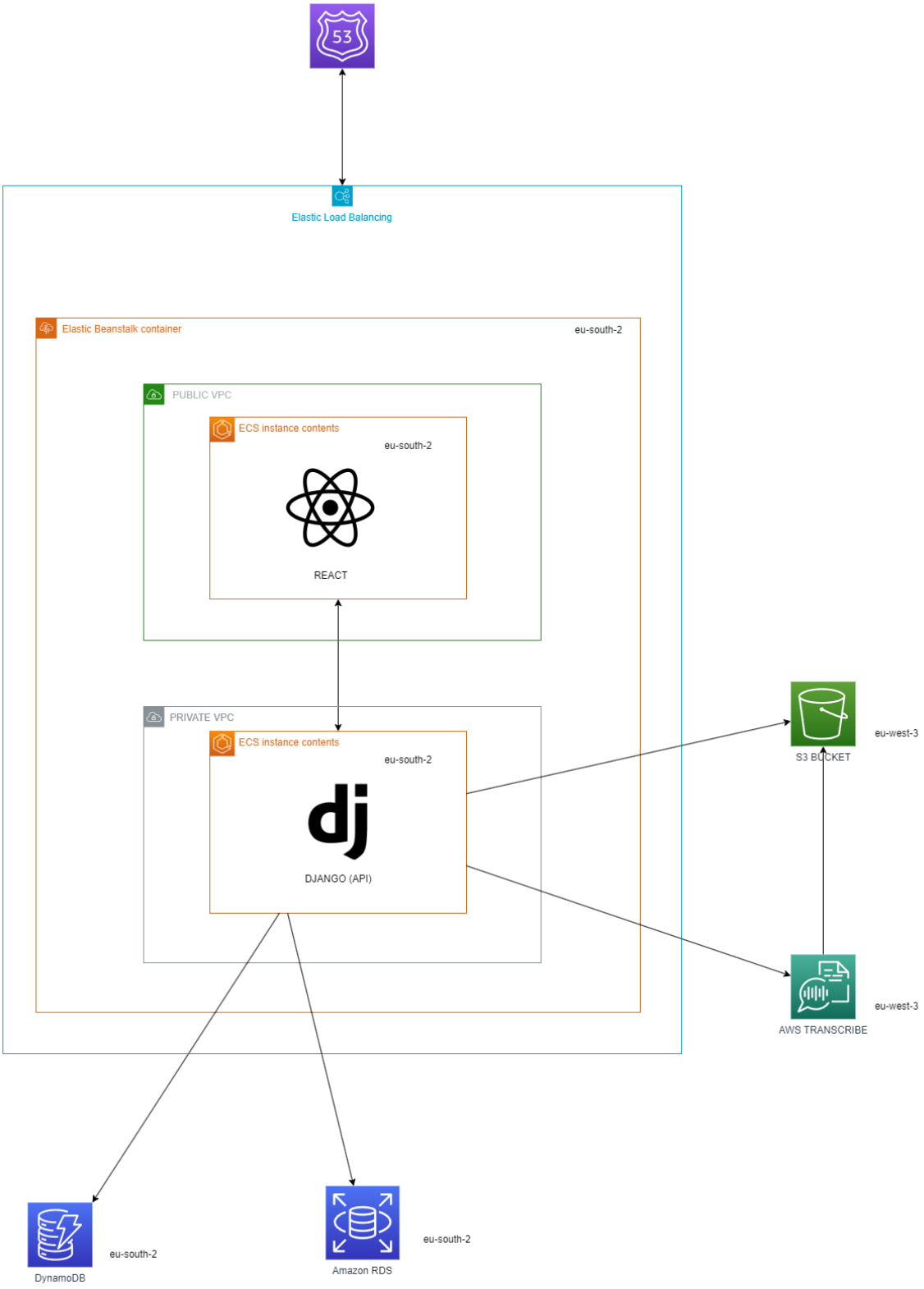
SQL Injection, y Clickjacking. Esto es de vital importancia para una organización como la Cruz Roja, donde la seguridad de los datos es fundamental.

4.3. **Desarrollo rápido:** Django sigue el principio DRY (Don't Repeat Yourself), que promueve la reutilización de código y minimiza la redundancia. Esto, junto con una serie de componentes reutilizables que Django proporciona "out of the box", puede acelerar significativamente el tiempo de desarrollo.

4.4. **Compatibilidad con bases de datos:** Django es compatible con una amplia variedad de sistemas de bases de datos, lo que permite una fácil integración con los sistemas de almacenamiento de datos existentes de la Cruz Roja.

4.5. **Django REST framework**: Este es un potente y flexible kit de herramientas que hace que sea fácil construir APIs Web. Proporciona funcionalidades como la serialización, la autenticación y la autorización, las vistas y los enrutadores, que pueden ayudar a crear APIs robustas y seguras.

Diagrama de la arquitectura



Otras tecnologías de transcripción

Deepgram y Transcribe son dos herramientas de reconocimiento de voz que permiten convertir audio en texto. Ambas herramientas tienen características útiles y pueden ser efectivas en diferentes situaciones, dependiendo de las necesidades del usuario.

Sin embargo, hay algunas razones por las que podrías considerar que Deepgram es mejor que Transcribe:

1. **Mayor precisión**: Deepgram utiliza una tecnología de reconocimiento de voz de vanguardia basada en inteligencia artificial, que le permite obtener una precisión de hasta el 99%. En comparación, la precisión de Transcribe varía entre el 85% y el 95%.
2. **Mayor velocidad**: Deepgram es capaz de transcribir audio en tiempo real, lo que significa que el texto aparecerá en pantalla mientras hablas. Transcribe puede tomar varios minutos para procesar y transcribir un archivo de audio.
3. **Mayor flexibilidad**: Deepgram permite personalizar la herramienta según las necesidades de cada usuario, lo que incluye la capacidad de entrenar el modelo para que reconozca acentos y vocabulario específicos. Transcribe tiene algunas opciones de personalización, pero no es tan flexible como Deepgram.
4. **Mayor seguridad**: Deepgram ofrece opciones de privacidad y seguridad para los datos del usuario, lo que incluye la capacidad de encriptar los datos en reposo y en tránsito, y la opción de utilizar servidores privados. Transcribe ofrece algunas opciones de seguridad, pero no es tan completo como Deepgram.

Descripción Técnica BETA

El proyecto está compuesto como se ha comentado anteriormente de dos módulos. El módulo de backend que va a ser una API en Django que va a permitir utilizar todas las funcionalidades para poder cargar audios, sacar estadísticas, obtener datos y visualizar esos mismos datos de diferentes maneras.

Django está limitado únicamente para que se pueda comunicar con el módulo de frontend, las dos bases de datos y todas las funcionalidades de Amazon, ya que se le ofrece un perfil de usuario IAM para poder utilizar los servicios de la cuenta.

Las funcionalidades que se van a incluir en el proyecto son los siguientes:

- Lo primero de todo un usuario debe tener un nombre y una contraseña para poder utilizar las funciones de la página.
- Una vez entra en la página dispone de varias pestañas en las que se van a incluir como mínimo información del usuario, estadísticas que se recojan de los análisis que ha hecho y poder utilizar todas las funciones que incluye la aplicación.
- Si lo que se quiere es grabar un audio, existe dos tipos de funciones que son subirlo desde tu propio ordenador o grabarlo en directo mediante la aplicación de React. Una vez está grabado se va a enviar el audio para poder procesarlo y obtener toda la información de él.

Flujo de ejecución del backend:

1. Se va a **cargar el audio en un cubo S3** el cual posteriormente después de haber hecho los análisis y si no se vuelve a utilizar el archivo en un tiempo superior a quince días se cambiará a otro tipo de cubo para reducir el coste.
2. Django va a **esperar a que la subida del audio se complete** y una vez ya se encuentre el archivo dentro del cubo, se va a proceder a **llamar a Amazon transcribe para que pueda analizar el audio y convertirlo a texto** para que sea almacenado posteriormente en Dynamodb.
3. Una vez se tenga una **primera transcripción del texto**, como muchas veces da errores, es necesario utilizar la inteligencia artificial para refinar la gramática y la coherencia del texto. Es por ello que se utiliza Chat GPT como inteligencia conversacional para poder **reescribir el texto y guardarlo de nuevo en la base de datos**.
4. Una vez se tiene una transcripción con coherencia se procede a aplicar algoritmos de inteligencia artificial para **responder a las preguntas contenidas dentro del texto y que son formuladas en la base de datos de la Cruz roja**.
5. Una vez terminado todo este proceso **se comienza con la estadística de los datos**. De esta manera se pueden sacar los grados de necesidad y las palabras más comunes que aparecen en el texto para después poder sacar otro tipo de análisis a gran escala que permite mejorar el enfoque de futuros proyectos que se hagan.
6. Cuando todo este proceso ha finalizado **se devuelve la información a React en formato JSON para que éste la pueda mostrar** de una manera interactiva y entendible por el ser humano.

Flujo de ejecución de front end:

1. Para la BETA únicamente se disponen de **tres pestañas a las que se va a poder navegar** la primera sería la de usuarios, la segunda serie para poder realizar las transcripciones de los audios de los atendidos y la tercera para poder ver las estadísticas y resultados que se han obtenido de esos audios.
2. **Usuarios**: en esta pestaña se va a contener toda la información del trabajador que está utilizando la aplicación en ese mismo momento. Para ello y las ventajas de utilizar una API es que esa **información se puede obtener desde la plataforma que se necesite y esté utilizando actualmente la Cruz Roja** para posteriormente poder mostrar la información que se quiera.
3. **SmartVoice**: en esta pestaña se podrá utilizar la máxima potencia que ofrece nuestro software. **Puedes grabar un audio en directo desde la propia aplicación de Riad o puedes subir un audio que hayas grabado desde whatsapp o cualquier otro tipo de dispositivo**. Simplemente tendrás que seleccionar el atendido que ya se tendrán guardados sus datos o se podría añadir un formulario en el que se añadan los datos del atendido para posteriormente poder subir el audio o grabar el audio.
4. **Estadísticas**: en esta sección **se van a mostrar todos los datos recolectados por la BETA que se ha creado para la Cruz Roja**. En un futuro estos datos se pueden modificar, mejorar y precisar de la manera que sea necesaria para obtener los mejores resultados según las necesidades que tenga la Cruz Roja.

La interfaz es una pequeña muestra de todo lo que puede ofrecer React de una manera muy sencilla y puede servir como base para saber todo lo que se puede construir en un futuro si se le dedica más tiempo y esfuerzo.

Funciones principales BETA

Este es un código de un proyecto de la Cruz Roja que consta de varias funciones. El objetivo del proyecto es utilizar tecnología para ayudar a las personas que llaman a la Cruz Roja en busca de ayuda.

La función "url_list" se llama cuando un usuario desea obtener una transcripción de un archivo de audio. Esta función carga el archivo de audio desde Amazon S3, luego utiliza AWS Transcribe para convertir el audio en texto. A continuación, utiliza una API de IA de OpenAI para corregir la gramática y ortografía del texto. Finalmente, almacena la conversación en una base de datos DynamoDB y devuelve la transcripción al usuario.

La función "lista_estadisticas" se utiliza para obtener estadísticas de las conversaciones almacenadas en la base de datos. Si se proporciona un correo electrónico, esta función devuelve solo las estadísticas para el correo electrónico proporcionado.

La función "lista_conversaciones" se utiliza para obtener todas las conversaciones almacenadas en la base de datos. Si se proporciona un correo electrónico, esta función

devuelve solo las conversaciones en las que el correo electrónico proporcionado aparece en el campo "atendido".

La función "cargar_usuarios_prueba" se utiliza para cargar usuarios de prueba en la base de datos. Devuelve un diccionario de usuarios cargados.

La función "pruebas_locas" se utiliza para borrar todas las estadísticas almacenadas en la base de datos.

La función "cargar_audio" se llama cuando un usuario desea cargar un archivo de audio. Esta función carga el archivo de audio desde el navegador del usuario y lo almacena en Amazon S3. A continuación, utiliza AWS Transcribe para convertir el audio en texto. A continuación, utiliza una API de IA de OpenAI para corregir la gramática y ortografía del texto y generar estadísticas sobre el texto. Finalmente, almacena la conversación en una base de datos DynamoDB y devuelve la transcripción y las estadísticas al usuario.

La función "login" se utiliza para permitir que los usuarios inicien sesión en el sistema. Devuelve un token de autenticación si las credenciales son correctas.

Las funciones "lista_atendidos", "lista_usuarios" y "buscar_atendido" se utilizan para obtener información de los usuarios almacenados en la base de datos.

En resumen, el proyecto utiliza tecnología para mejorar la calidad de las conversaciones entre los usuarios y la Cruz Roja. Utiliza AWS Transcribe y una API de IA de OpenAI para mejorar la precisión de las transcripciones y la corrección gramatical del texto, y almacena las conversaciones en una base de datos DynamoDB para permitir el análisis de estadísticas.

Flujo de ejecución buscado

1. La aplicación web debe tener una interfaz que permita a los operarios de la Cruz Roja iniciar y mantener una conversación con las personas desfavorecidas. La interfaz también debe incluir una sección de formulario donde los operarios puedan seleccionar y completar los campos requeridos.
2. Cuando los operarios inicien una conversación, la aplicación web debe enviar la información al servidor Django a través de una solicitud HTTPS.
3. El servidor Django debe recibir la solicitud y enviarla a la cola de SQS de Amazon para que se procese en orden secuencial.
4. El procesamiento de la solicitud se puede realizar mediante AWS Lambda, que puede escuchar la cola de SQS y procesar cada solicitud de conversación en el orden en que se reciben.
5. AWS Lambda puede llamar a la API de Transcribe de AWS para transcribir la conversación de voz a texto. Luego, el texto resultante se puede procesar con la lógica de negocios de la aplicación y almacenarse en una base de datos de Amazon DynamoDB.
6. Si se requiere alguna acción adicional, como el envío de notificaciones o la actualización de otras aplicaciones, AWS Lambda puede enviar solicitudes HTTP a otros servicios de AWS como Amazon SNS o AWS Lambda.
7. Finalmente, AWS Lambda puede enviar una respuesta HTTP con la información del formulario y cualquier otra información relevante al servidor Django para que se muestre a los operarios de la Cruz Roja.

Costes y estructura

Los costos de los servicios de AWS pueden variar en función de una serie de factores, incluyendo la región en la que te encuentras, el tiempo de uso y los precios actuales. Basandonos en una actividad de unos 10.000 usuarios al mes.

- **ECS (Elastic Container Service)** para React:Los costos de ECS dependen de los recursos de EC2 que utilices para ejecutar tus contenedores. Sin embargo, ECS no tiene costo adicional, solo pagas por los recursos que consumes.
- **EC2 para Django**:Una instancia t3.medium (2 vCPUs, 4 GB de memoria) en la región de EE.UU. Este (N. Virginia) cuesta aproximadamente $0.0416 por hora, lo que suma alrededor de $30 al mes.
- **DynamoDB**:Los costos de DynamoDB dependen del rendimiento de lectura y escritura que necesites. Si utilizas la capacidad de lectura y escritura a petición, los costos pueden ser muy bajos para un uso moderado. Por ejemplo, para 25 unidades de lectura y escritura, el costo sería de aproximadamente $14.50 al mes.
- **RDS** (small):Una instancia db.t3.small en la región de EE.UU. Este (N. Virginia) cuesta aproximadamente $0.026 por hora, lo que suma alrededor de $19 al mes.
- **Transcribe**:Los primeros 60 minutos al mes son gratuitos. Después de eso, cuesta $0.024 por minuto. Si asumimos que transcribes una hora de audio al día, eso sería aproximadamente $22.32 al mes.
- **S3 (20 GB)**:Los primeros 50 TB al mes cuestan $0.023 por GB, por lo que 20 GB te costaría alrededor de $0.46 al mes.

Por lo tanto, los costos totales de los servicios de AWS mencionados anteriormente serían de alrededor de $86.28 al mes, sin tener en cuenta otros costos como la transferencia de datos, las operaciones de S3 y los costos adicionales por el uso intensivo de otros servicios. Además, AWS puede ofrecer ahorros si reservas instancias por un período de tiempo más largo.

Links de referencia

GitHub: https://github.com/matahaxusen/cruzroja
Video BETA: https://youtu.be/WNVg7tP4mgg