Instituto Tecnológico y de Estudios Superiores de Monterrey

Arquitectura de Software

Fecha: 17 de junio de 2023

Desarrollo e implantación de sistemas de software

Alumnos:

Diego Alonso Bugarin Estrada A01620485
Edgar Alexandro Castillo Palacios A00830568
Hugo Edgar Palomares Estrella A01741537
Santiago Ortiz Pérez A01620647

El presente documento tiene como finalidad el presentar la arquitectura de software por el cual está compuesto el programa web *WizeTalk*, abarcando la arquitectura lógica, de procesos, de desarrollo y física que constituyen a este proyecto. Para esto fue importante realizar un análisis de la arquitectura monolítica y de microservicios para determinar aquella que se adapte de mejor manera a la naturaleza del proyecto, abarcando así las funcionalidades y servicios de manera eficiente. A continuación, se van a presentar las propiedades de cada arquitectura para entender el motivo por el cual la arquitectura seleccionada fue la más adecuada en el desarrollo del proyecto.

La arquitectura monolítica es aquella que integra distintos componentes en un solo programa en una sola plataforma. Los componentes que pueden ser parte de esta arquitectura son la autorización (login), presentación (HTML), lógica del negocio, base de datos e integración de la aplicación (REST API). El desarrollo monolítico simplifica el desarrollo de software al ser desarrollado en un código base, así mismo facilita la ejecución de las pruebas y el despliegue del producto final. Sin embargo, el hecho de estar desarrollado en la misma base implica enfrentar a problemas de mantenimiento en casos de desarrollos extensos y complejos, así mismo la escalabilidad de la aplicación puede ser un obstáculo cuando los distintos módulos empleados tienen problemas con el uso de recursos, finalmente el hecho de estar compartiendo los mismos recursos implica que en caso de falla de alguno de ellos puede provocar el colapso de todos los procesos de la aplicación. (Haq, S., 2018)

Por otra parte, se encuentra la arquitectura de microservicios, la cual tiene la intención de desarrollar una aplicación por medio de su descomposición en bloques. Cada uno de los módulos identificados tiene como finalidad el cubrir una de las metas del negocio mientras busca integrar una interfaz de comunicación simple con los otros servicios. Una parte esencial de esta arquitectura es que cada uno de los módulos cuenta con su propia base de datos, buscando reducir la dependencia entre cada uno de los servicios. Con esta arquitectura se asegura la entrega y despliegue constante de aplicaciones de mayor complejidad, así mismo, mejora el proceso de pruebas al ser modular, de la misma forma la organización y desarrollo de la aplicación es más eficiente al ya estar segmentado y al no ser completamente dependiente de otro módulo. Sin embargo, dentro de las desventajas de esta arquitectura se encuentra la tarea de diseñar aplicaciones modulares más complejas, así mismo, el desarrollo de los canales de comunicación entre los servicios. (Haq, S., 2018)

Tomando en cuenta las definiciones y usos de las dos arquitecturas mostradas anteriormente, hemos optado por implementar en el proyecto la arquitectura monolítica, la decisión se argumenta a partir de la facilidad en su diseño, desarrollo e implementación dada a la restricción de tiempo que se tiene y el número de integrantes que componen al equipo. Otro aspecto que influyó en la toma de esta decisión fue que todos los miembros han tenido experiencia previa en el diseño y desarrollo de programas monolíticos, por lo que la implementación de una arquitectura de microservicios representaría la investigación y capacitación de cada miembro, abriendo la posibilidad de presentar más errores en su implementación, restando tiempo productivo de desarrollo y testeo. Finalmente consideramos que el producto final no cuenta con un alto grado de complejidad, por lo que aspectos como

el mantenimiento, desarrollo de pruebas y despliegue del programa no representan un aspecto determinante para optar por la arquitectura de microservicios.

Objetivos a Nivel Arquitectura y Restricciones

El producto, al tratarse de una propuesta a una solución identificada por la empresa de Wizeline, tiene como prioridad la implementación y desarrollo de un programa funcional que refleje las funcionalidades y alcances de este sistema para brindar una solución viable al problema presentado. Por lo que aspectos como la escalabilidad, seguridad y diseño de la propuesta pasan a estar en un segundo plano de acuerdo con el cliente. Por lo tanto, los objetivos que se tienen respecto al producto final son los siguientes:

· Entrega de un programa funcional en tiempo y forma conforme a la fecha de entrega planteada por el cliente.

· Diseño e implementación de una arquitectura de software sencilla para el desarrollo del producto.

· Ejecución de pruebas de componentes y E2E de manera sencilla.

· Facilidad en la corrección de errores de integración e implementación.

Una vez se hayan identificado el tipo de arquitectura a implementar en el producto, así como el framework que mejor se adecúe a los objetivos del cliente (en este caso se emplea Remix), es necesario identificar aquellas restricciones presentadas en ambas partes para tomarlas a consideración en el desarrollo del producto y abatirse de manera puntual.

Remix es un nuevo framework para el desarrollo de aplicaciones web que prioriza el rendering del lado del servidor, así mismo hace uso de rutas anidadas. A continuación, se van a presentar las restricciones presentes ante el uso de este framework:

- El uso de rutas anidadas representa un desafío para compartir información entre páginas, por lo que cada una de las rutas tiene que contar con un “handle” que estará disponible en el hook “useMatches” en el nivel más alto.

- Al no contar con un rendering del lado del cliente el número de queries se eleva al ser necesario extraer información en cada ruta para ser desplegada en la interfaz de usuario.

- Compartir información entre los componentes de la UI es limitada al ser necesario su procesamiento en el lado del servidor, por lo que puede ser más lento este proceso en contraste a otros frameworks.

Se estarán utilizando los servicios de OpenAI para la integración de ChatGPT y Whisper en el producto final. Al ser modelos de AI que están en constante mejora, estos

servicios están en constante actualización, por lo que es necesario estar al pendiente de estas mejoras para su incorporación en el proyecto.

El uso de los servicios de AWS y de OpenAI son necesarios para alcanzar a cumplir las metas propuestas por el cliente. Ante esta situación se analizó el uso de los SDKs de cada una de las empresas y las ventajas o desventajas respecto al uso de los métodos fetch para extraer la información necesaria para el funcionamiento de la aplicación. Tras un análisis de ambos métodos y las sugerencias dadas por el cliente, se optó por el empleo de este último método al no depender de las actualizaciones de las librerías y por ende, minimizar los errores provocados por la incompatibilidad de las versiones.

Arquitectura Lógica

Es importante entender la arquitectura por la que va a estar compuesto nuestro producto final desde un punto de vista lógico. Para esto ha sido necesario identificar el funcionamiento de la aplicación y los niveles por el cual va a estar compuesto.

Los niveles que van a conformar la solución propuesta son los siguientes:

- Cliente: Son aquellas aplicaciones empleadas por el usuario para acceder a los servicios del portal. Depende de los tipos de dispositivos y buscadores que van a ser empleados por el cliente. En el caso del presente proyecto se contempla el despliegue del producto en un navegador web desde cualquier máquina personal o de escritorio.

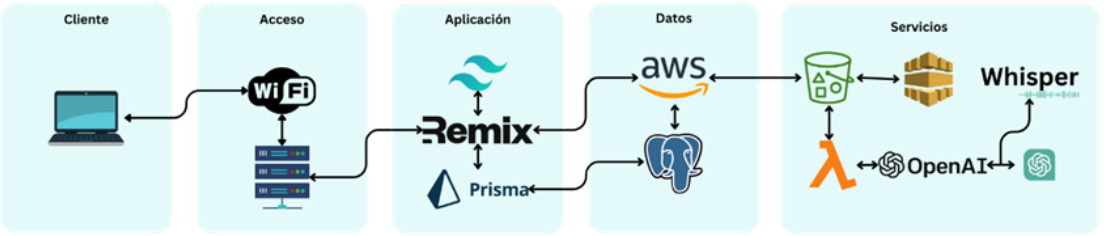
- Acceso: Establecen los métodos de ingreso a la red del sistema. Esto se va a lograr a través del empleo del internet para permitir la conexión entre los dispositivos de los trabajadores con los servidores de la aplicación y los servicios de AWS y OpenAI que serán empleados.

- Aplicación: Provee el acceso del usuario al sistema para el despliegue de la información y satisfacer los requerimientos del cliente. Para esto se integra la parte de diseño del producto con su funcionalidad. Es por este motivo que se emplea el framework de Remix que permitirá esta integración, por el otro lado, la librería de Tailwind apoyará a desarrollar de manera más sencilla y eficiente el frontend para el despliegue de esta información.

- Datos: Consiste en aquellos datos implicados en la aplicación, abarcando desde su almacenamiento y extracción para ser presentados ante el usuario. Para este nivel se optó por emplear una base de datos relacional a petición del cliente, por el cual se implementó PostreSQL.

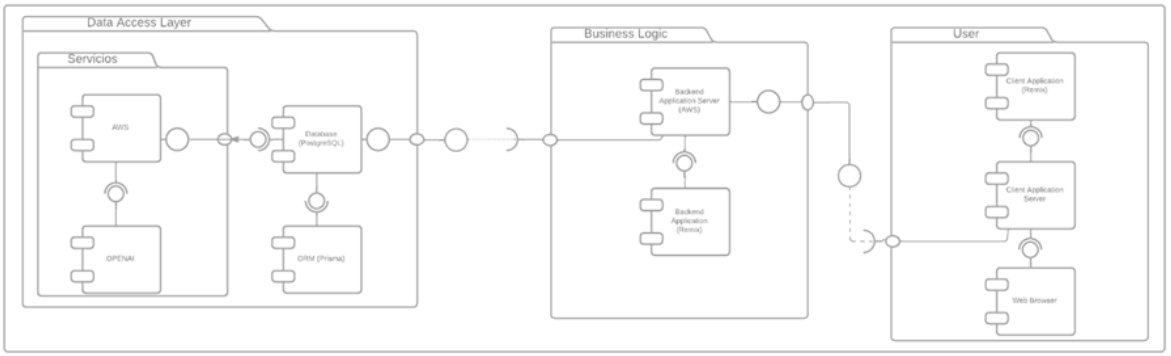
- Servicios: Implican todos los productos que serán implementados en el software con el fin de satisfacer los requerimientos especificados por el

cliente. Estos servicios abarcan la transcripción de los videos por medio de Whisper, la generación de calificaciones empleando ChatGPT, el uso de la base de datos alojada en RDS de AWS, y el almacenamiento en la nube con S3.



Arquitectura de Desarrollo

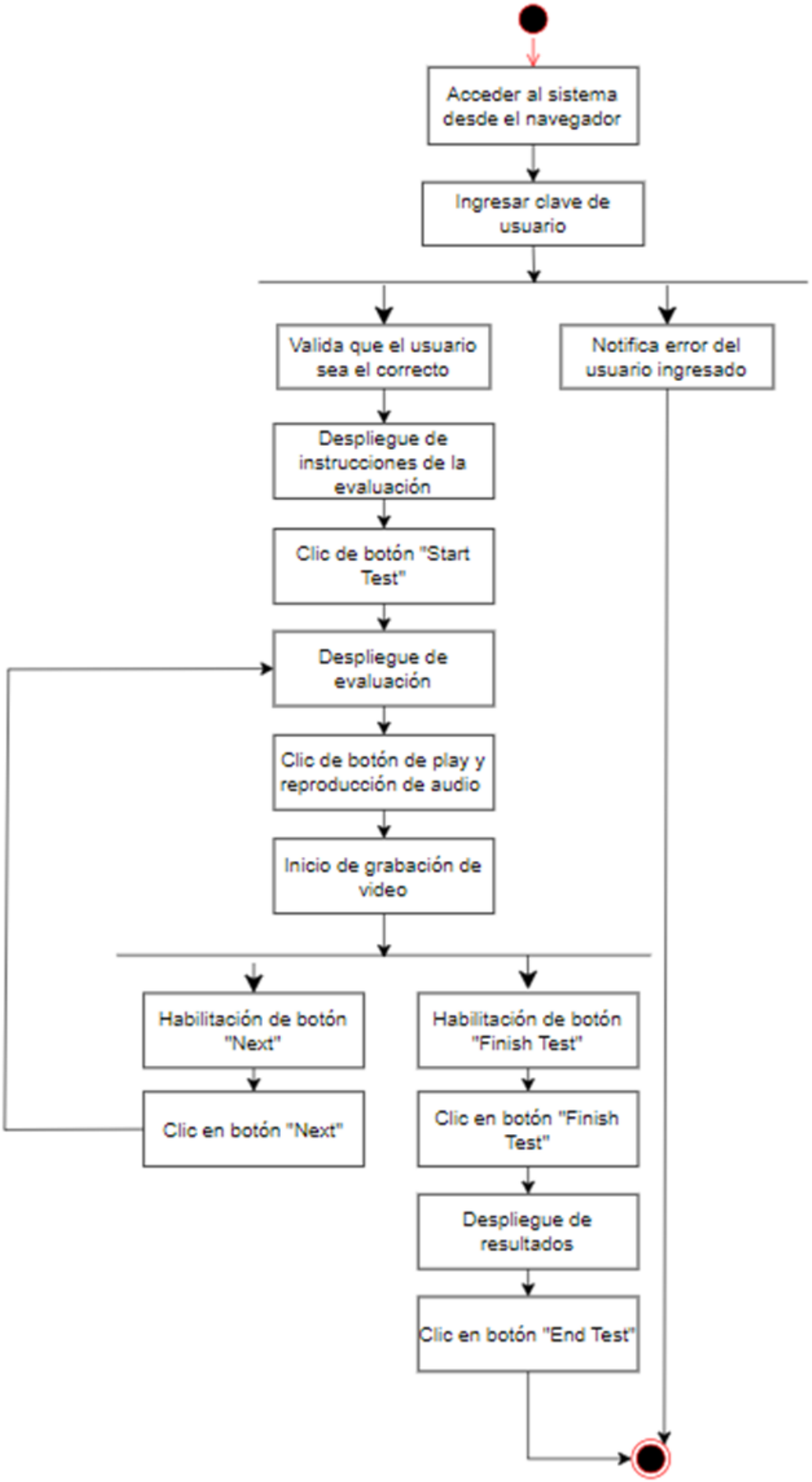
Se identificó el proceso involucrado en el desarrollo del sistema, el cual establece las tecnologías implementadas para la aplicación web de WizeTalk. Aquí están implicadas 4 capas, el de usuario que equivale al de presentación establecido en la arquitectura lógica, de lógica empresarial y el de acceso a datos donde también se abarcan los aspectos de los usos de servicios por medio de los APIs de los proveedores. Esta composición se puede presenciar de mejor manera en el siguiente diagrama de componentes:



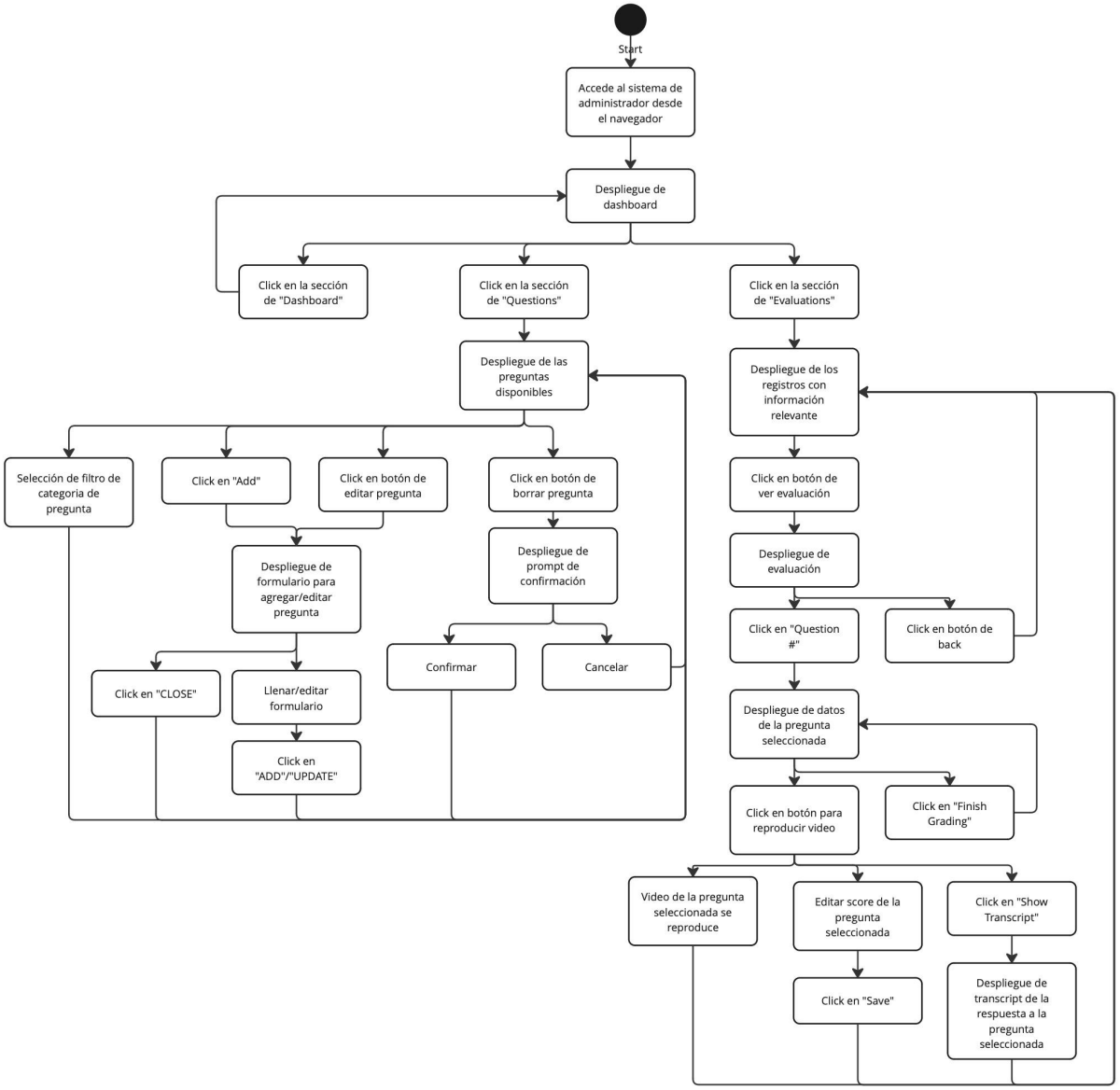
Arquitectura de Procesos

Se identificaron los procesos que van a tomar lugar en el producto final, desde las funcionalidades que va a realizar el usuario para hacer su evaluación hasta aquellas facultades que tendrán los administradores. En total se identificaron 4 procesos, la evaluación del usuario, despliegue de dashboard, modificación de calificaciones de las evaluaciones y adición de preguntas.

A continuación, se explicará el proceso realizado por el usuario para realizar su evaluación de nivel de inglés y de conocimientos tecnológicos:

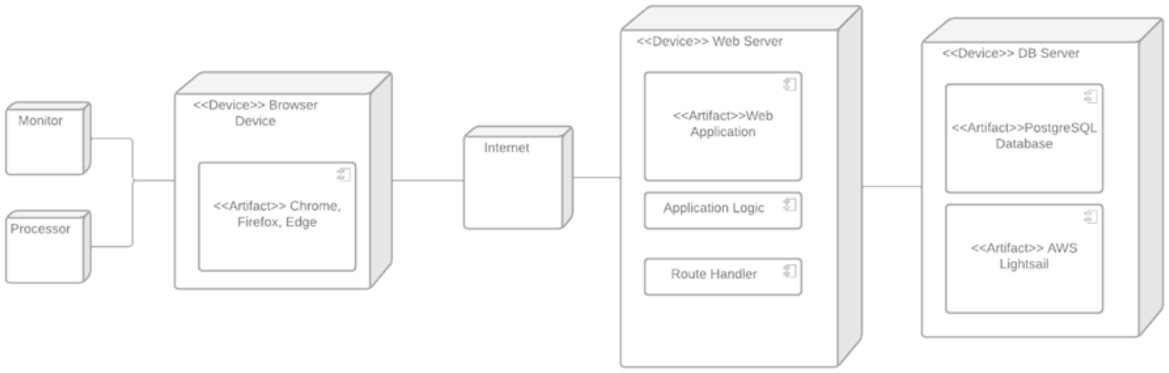

Acceder al sistema desde el navegador
Ingresar clave de usuario
Valida que el usuario sea el correcto
Notifica error del usuario ingresado
Despliegue de instrucciones de la evaluación
Clic de botón "Start Test"
Despliegue de evaluación
Clic de botón de play y reproducción de audio
Inicio de grabación de video
Habilitación de botón "Next"
Habilitación de botón "Finish Test"
Clic en botón "Next"
Clic en botón "Finish Test"
Despliegue de resultados
Clic en botón "End Test"

El siguiente diagrama representa los procesos que puede realizar un administrador en la plataforma, que abarca tanto visualización como edición de resultados y preguntas, así como despliegue de gráficas con información relevante para el administrador.



Arquitectura Física

Se identificaron aquellos elementos de la arquitectura de WizeTalk que conforman a la parte física del sistema, es decir, aquellos dispositivos físicos y reales que estarán encargados de ejecutar todas las funcionalidades especificadas en el sistema. Para esto se detectaron 3 componentes que serán empleados, el primero de ellos es el navegador que tendrá que utilizar el usuario para ingresar a la plataforma, el segundo es el servidor web en donde se aloja la aplicación y finalmente están aquellos servidores que contendrán la base de datos. Esto se puede entender de mejor manera con la siguiente figura:



Bibliografía

Haq, S., (2018). Introduction to Monolithic Architecture and MicroServices Architecture. Consultado el 15 de junio de 2023, de https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63