**UNIVERSIDAD PRIVADA DEL VALLE**

**FACULTAD DE INGENIERÍA**

**INGENIERÍA DE SISTEMAS INFORMÁTICOS**

**UNIVALLE – COCHABAMBA**

**PROYECTO DE SISTEMAS**

**TOTEM**

**DOCENTE:** Gaston Silva Sanchez

**Cochabamba octubre de 2022**

**Gestión 2-2**

**Manual Técnico**

**Equipo: Tótem**

1. **Introducción.**

El proyecto Tótem es una innovadora solución diseñada para facilitar la ubicación de lugares deseados de manera rápida y sencilla. Consta de un tótem interactivo equipado con una pantalla, un botón y un micrófono. Al presionar el botón, los usuarios podrán utilizar comandos de voz para indicar la ubicación que desean encontrar. El tótem realizará búsquedas en línea y presentará los resultados en la pantalla, mostrando el nombre del lugar, un carrusel de imágenes relacionadas, una descripción detallada y un mapa con la ruta para llegar al destino.

Este proyecto tiene como objetivo principal ayudar a las personas a encontrar fácilmente los lugares a los que desean llegar, brindándoles información completa y visualmente atractiva. Además, busca simplificar el proceso de navegación y eliminar la necesidad de realizar múltiples consultas en dispositivos móviles o computadoras. Con el tótem, la ubicación deseada estará al alcance de la mano, proporcionando una experiencia intuitiva y eficiente para los usuarios en diversos ámbitos, como turismo, navegación en interiores o simplemente descubrir nuevos lugares en su ciudad.

1. **Descripción de proyecto.**

El proyecto Tótem se desarrollará utilizando una combinación de hardware y software para lograr su funcionalidad integral. En cuanto al hardware, se utilizará una pantalla de alta resolución para mostrar los resultados de búsqueda y la interfaz de usuario. Además, se empleará un micrófono para capturar los comandos de voz de los usuarios y un botón para activar el proceso de búsqueda. Todo este hardware estará conectado a un dispositivo de procesamiento, una Raspberry y una computadora, que se encargará de controlar las interacciones y ejecutar el software.

En cuanto al software, se utilizarán diversas técnicas y herramientas para lograr las funcionalidades clave del proyecto como: React, SQL Server, visual Studio, visual studio Code, Azure.

La presentación de los resultados se llevará a cabo mediante la combinación de elementos visuales y textuales en la pantalla. El nombre del lugar se mostrará de manera destacada, seguido de una descripción detallada que proporcionará información adicional sobre el lugar. Asimismo, se incluirá un mapa con la ruta para llegar al destino, el cual se generará utilizando una API de mapas y se mostrará en la pantalla del tótem.

1. **Roles / integrantes.**

|  |  |
| --- | --- |
| Integrantes | Rol |
| Alan Abdel Montaño Cruz | Team Leader |
| Luciana Mayra Blanco Aranibar | Integration |
| Nicolas Tadeo Meneses Barbery | DB Architect / infrastructure |
| Alicia Natalia Virreira Idina | Scrum Master |
| Fernando Daniel Aparicio Estrada | GIT Master |

1. **Arquitectura del software.**

La arquitectura de nuestro proyecto se compone de varios componentes clave que interactúan entre sí para lograr nuestros objetivos. A continuación, presento una explicación de la estructura y organización del software, considerando los elementos mencionados:

Componentes principales:

En nuestro proyecto utilizamos una base de datos SQL Server para almacenar y gestionar los datos. La base de datos actúa como una fuente centralizada de información. Utilizamos el modelo relacional de SQL Server para definir la estructura de la base de datos y establecer relaciones entre las tablas.

Para la comunicación entre el frontend y la base de datos, desarrollamos una API utilizando ASP.NET Core. Esta API proporciona puntos de acceso para interactuar con los datos almacenados en la base de datos. Definimos controladores y rutas en la API para recibir solicitudes desde el frontend y realizar operaciones correspondientes en la base de datos.

La interfaz de usuario de nuestro proyecto se implementó utilizando React, un framework de JavaScript. React nos permite construir una interfaz de usuario interactiva y receptiva, proporcionando componentes reutilizables y una gestión eficiente del estado de la aplicación. Nuestro frontend se comunica con la API para obtener y enviar datos al backend.

Interacciones entre componentes:

Nuestro frontend desarrollado con React se comunica con la API utilizando solicitudes HTTP. Utilizamos bibliotecas como Axios para realizar llamadas a la API y enviar datos a través de métodos como GET, POST, PUT o DELETE. Recibimos las respuestas de la API en formato JSON y las utilizamos para actualizar la interfaz de usuario según sea necesario.

La API establece una conexión con la base de datos SQL Server para realizar operaciones de lectura y escritura. Utilizamos consultas SQL para recuperar datos de la base de datos o modificarlos según las solicitudes recibidas desde el frontend. Para simplificar las operaciones de acceso a la base de datos y mapear objetos a tablas y viceversa, utilizamos un ORM (Object-Relational Mapping) como Entity Framework Core.

Patrones de diseño:

Para organizar la estructura de nuestra aplicación, seguimos el enfoque de arquitectura de tres capas. Tenemos una capa de presentación (React frontend), una capa de lógica de negocio y una capa de acceso a datos (API y base de datos SQL Server). Este enfoque facilita la escalabilidad y el mantenimiento del sistema.

En el desarrollo de la API, aplicamos el patrón de diseño MVC (Modelo-Vista-Controlador) para separar la lógica de negocio de la presentación y el control de las solicitudes. También utilizamos el patrón Repository para abstraer el acceso a la base de datos y facilitar las operaciones CRUD.

Además de estos aspectos, debemos tener en cuenta consideraciones adicionales, como la seguridad de la aplicación, el despliegue y la escalabilidad en Azure. Utilizamos servicios de Azure como Azure App Service para alojar la API y Azure SQL Database para hospedar la base de datos SQL Server. Además, nos aseguramos de implementar las medidas de seguridad necesarias para proteger nuestros datos y recursos.

1. **Requisitos del sistema.**

**Requerimientos de Hardware (mínimo):** 500 MB de almacenamiento, Intel Pentium 4, 512 md de RAM.

**Requerimientos de Software: (cliente):**

Windows 7, Azure Portal, Visual studio Code 2019.

**Requerimientos de Hardware (server/ hosting/BD)**

Procesador 1.4 GHz (64 bits), RAM 512 MB, Disco duro 32 GB.

**Requerimientos de Software (server/ hosting/BD)**

Azure Portal para todo

1. **Instalación y configuración.**

En este proyecto, la mayoría de los componentes están alojados en Azure Portal. Esto incluye la API, la base de datos con su respectivo servidor y el storage de imágenes. Si utilizamos la misma cuenta de Azure de Tótem, no será necesario realizar ningún cambio, ya que el proyecto de React que tiene que descargar ya está configurado con las direcciones correctas. Sin embargo, si cambiamos de cuenta, será necesario alojar nuevamente la base de datos, la API y el storage de imágenes. Luego, en el proyecto de React, deberemos modificar las URL o direcciones que se conectan con la API, y también cambiar la cadena de conexión en la API para establecer la conexión tanto con la base de datos como con el storage de imágenes, una vez realizado todo eso debemos agregar las librerías en el proyecto de React las cuales son las siguientes:

**Instalación de la dependencia Node.js**

768 install

**Instalación de la librería de mandado de emails**  
npm install @emailjs/browser --save

**Instalación de la librería Sweet Alert para el uso de alertas**

npm install --save sweetalert2 sweetalert2-react-content

**Instalación de la librería de cifrado de contraseñas**  
*npm install crypto-js*

**Instalacion de librería de axios**

npm install axios

Estas son las configuraciones que deberemos realizar para poder utilizar u instalar nuestra aplicación.

1. **Procedimiento de hosteado / hosting.**

**Pasos para hostear la Base de datos en Azure:**

Para el host de la base de datos usamos Azure Portal en el cual creamos un base de datos y un servidor al cual la base de datos se conecta, primero debemos ingresar a nuestra cuenta de Azure, en el buscador de Azure debemos buscar “Azure SQL” nos enviará a una nueva ventana y en la nueva ventana no aparecerá todos los servidores y bases de datos que tengamos, además de un botón de crear que es el botón que nosotros necesitamos una vez apretemos ese botón, nos llevara a una nueva ventana con tres posibles opciones de lo que queremos crear nosotros escogemos la opción de crear “SQL Database” una vez apretemos el botón tendremos que completar los requisitos que nos pide: El de la suscripción que normalmente se pone automáticamente, el grupo de recursos que si no tenemos unos tendremos que crear uno, El nombre de la base de datos, El servidor que si no tenemos uno tenemos que crearlo al momento de crear este servido es recomendable que al colocar la ubicación se coloque la de (US), igual dentro de la creación del servidor el método de autenticación escojamos la segunda opción “Uso de la autenticación de SQL y Azure AD”, igualmente al momento de crear el servidor al momento de establecer administrador de Azure AD seleccionemos la primera opción que no aparezca, ahora si volveremos a la ventana de creación de la base de datos SQL en el apartado de proceso y almacenamiento se debe de seleccionar la más básica así el precio será mucho más económico, por ultimo em redundancia del almacenamiento de copias de seguridad escogemos la primera opción. Después ya podemos apretar el botón

de revisar y crear para que empiece la creación de nuestra base de datos y nuestro servidor, para poder hacer cambios en la base de datos debes dirigirte al servidor de la base de datos ingresar en el para después ir en el menú izquierdo a la sección de seguridad y entrar en redes, marcar la opción de redes seleccionadas y después agregar tu IP en la parte de abajo la cual se hace automáticamente tui solo tienes que poner agregar IP y se agregara solo, ahora si podemos hacer cambios en la base de datos.

**Ahora estos son los pasos para crear el storage de imágenes:**  
1.1 Crear un contenedor de blobs Una vez que hayas creado la cuenta de almacenamiento, crea un nuevo contenedor de blobs dentro de la cuenta. Establece un nombre único para el contenedor y define las opciones de acceso público o privado según tus requisitos de seguridad.

1.2 Obtener las credenciales de acceso Para poder interactuar con Azure Blob Storage desde tu proyecto, necesitarás obtener las credenciales de acceso. Estas credenciales incluyen la cadena de conexión a la cuenta de almacenamiento, que contiene la información necesaria para autenticarse y autorizar las operaciones con los blobs.

1.2.1 Integración de Azure Blob Storage en tu proyecto Una vez que hayas configurado Azure Blob Storage, sigue los siguientes pasos para integrarlo en tu proyecto y utilizarlo para el manejo de imágenes:

2.1 Instalación del paquete NuGet Utiliza el administrador de paquetes NuGet de tu proyecto para instalar el paquete Microsoft.Azure.Storage.Blob. Este paquete proporciona las bibliotecas necesarias para interactuar con Azure Blob Storage desde tu código.

2.2 Configuración de la conexión En tu código, establece la conexión con Azure Blob Storage utilizando la cadena de conexión obtenida en el paso. Puedes utilizar la clase CloudStorageAccount para crear una instancia de la cuenta de almacenamiento utilizando la cadena de conexión.

2.3 Creación de un blob Para almacenar una imagen en Azure Blob Storage, crea un nuevo blob en el contenedor de blobs correspondiente. Puedes utilizar la clase CloudBlobClient para obtener una referencia al cliente de blobs y luego crear un nuevo blob utilizando la clase CloudBlockBlob. Proporciona un nombre único para el blob y establece los metadatos necesarios.

2.4 Subida de una imagen Una vez que hayas creado un blob, puedes subir una imagen a Azure Blob Storage. Utiliza el método UploadFromStream de la clase CloudBlockBlob para cargar la imagen desde un flujo de datos, como un FileStream o un MemoryStream. También puedes utilizar otros métodos proporcionados por la clase CloudBlockBlob según tus necesidades.

2.5. Obtención de la URL de una imagen Después de subir una imagen a Azure Blob Storage, puedes obtener la URL correspondiente al blob. Utiliza la propiedad Uri del objeto CloudBlockBlob para obtener la URL completa. Puedes utilizar esta URL en tu proyecto para mostrar la imagen o generar enlaces.

2.6. Descarga y eliminación de imágenes Para descargar una imagen desde Azure Blob Storage, utiliza métodos como DownloadToStream o DownloadToFile de la clase CloudBlockBlob. Estos métodos te permitirán obtener la imagen y almacén

**Para crear subir a API a la cuenta de Azure:**

primero se debe crear una web app con la suscripción:

Entramos a nuestra cuenta de Azure portal, ponemos la opción “Crear un Recurso” y seleccionamos la opción que dice Web App.  
Patrón de fondo

Descripción generada automáticamente con confianza baja

Luego llenamos los datos que nos piden. Primero seleccione la suscripción que queremos utilizar y luego el grupo de recurso que usaremos para nuestro proyecto y en el cual se encuentra la base de datos.  
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Le damos un nombre y en la parte que dice publicar seleccione la opción Code.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En donde dice Runtime stack seleccione la versión de .NET que se haya usado, en el caso de nuestra API es versión .NET 6.Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Seleccione el Pricing Plan que en nuestro caso seleccionamos el Básico.  
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

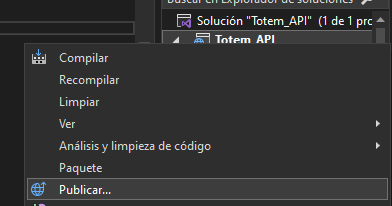
Descripción generada automáticamente

Y por último seleccione el botón de “Preview + créate” y el botón de crear.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ya tendríamos el Web app en el que subiremos nuestra API. El siguiente paso es entrar a visual studio 2022 y abrir el proyecto en el que se encuentra la API.  
En el Explorador de soluciones, haga clic con el botón derecho en el proyecto y seleccione Publicar.



En el cuadro de diálogo Publicar, seleccione Azure y seleccione el botón Siguiente.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Seleccione Azure App Service (Windows) y seleccione el botón Siguiente.

Interfaz de usuario gráfica, Texto, Aplicación

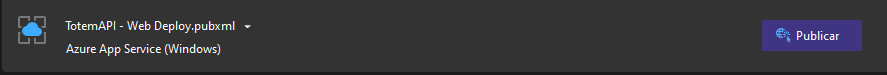
Descripción generada automáticamente

En la siguiente ventada debe iniciar sesión con la cuenta con la que haya creado el Web app, una vez haya iniciado sesión deberá seleccionar la suscripción que esta utilizando.  
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Al escoger la suscripción en un cuadro en la parte de abajo le aparecerá una carpeta con el nombre de su grupo de recurso, y dentro de ella el Web app que creo para el proyecto.

Selecciona el Web app y presiona el botón finalizar.

Al hacer esos pasos correctamente debió llevarlo a una ventana que en la parte superior tiene esta sección.  


Al seleccionar el botón publicar la API se subirá automáticamente al Web app creado. Una vez finalizado el proceso de publicación se abrirá un navegador con la dirección de la api ya en la nube y lista para consumir.

1. **GIT.**

Este es el link de repositorio de la API:

<https://github.com/Totem-Univalle/Totem_Api_Univalle.git>

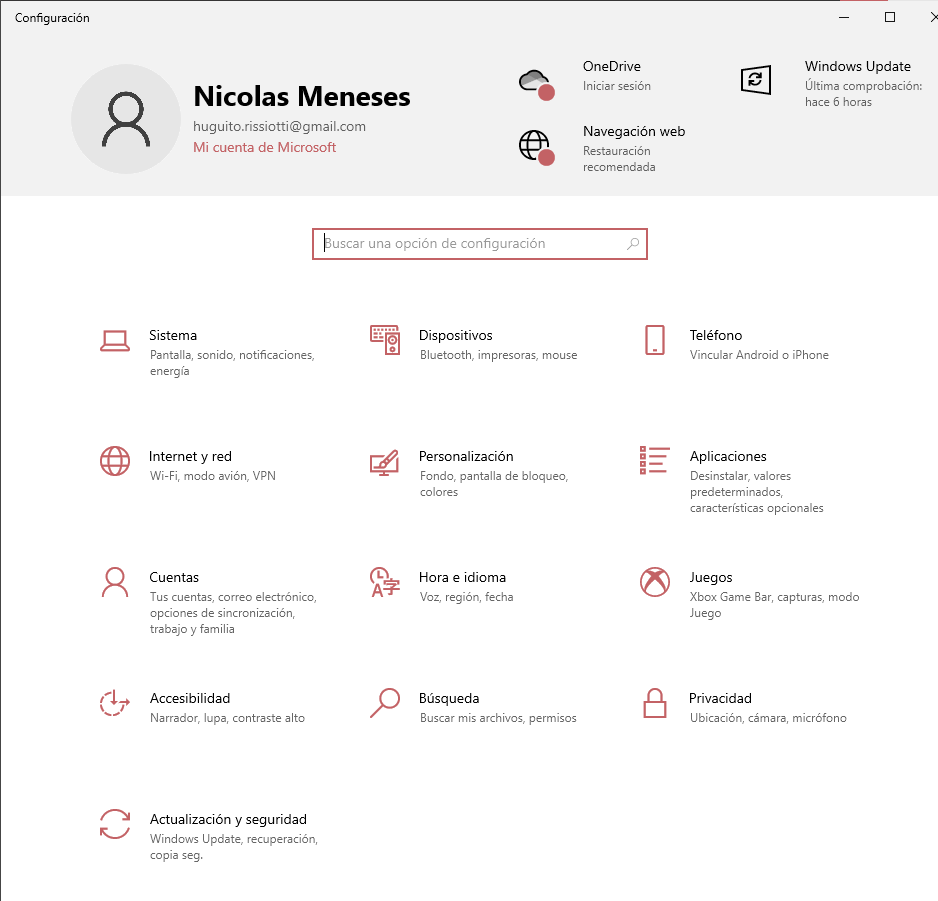
Este link es del proyecto:

<https://github.com/Totem-Univalle/Totem.git>

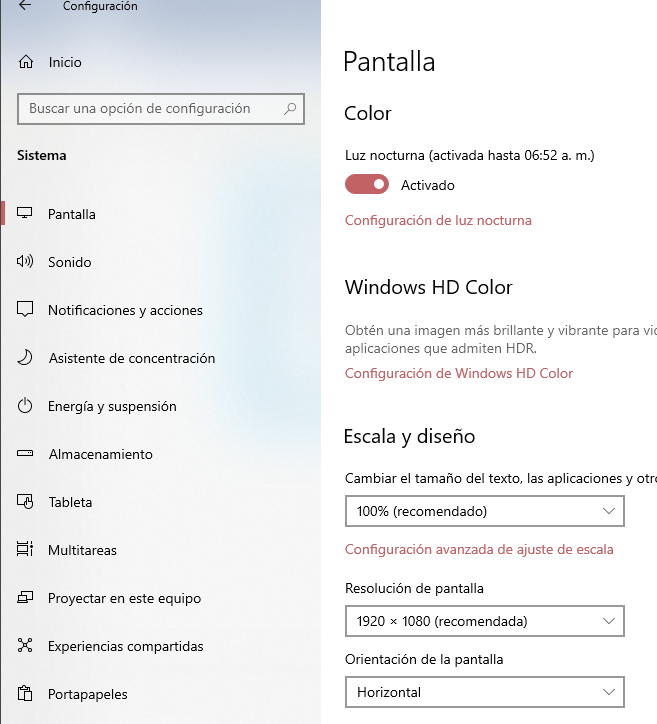
1. **Personalización y configuración.**

Lo único que la persona tiene que configurar es que vaya a Ajustes en la computadora

Y seleccione la opción de sistema.



Y en pantalla seleccionar Orientación de la pantalla seleccionamos la “opción de Vertical”



1. **Seguridad.**

El presente proyecto ha implementado un sistema de autenticación y obtención de información de una API mediante el uso de tokens y cookies. Estos mecanismos son indispensables para asegurar que solo los usuarios autorizados puedan acceder a la API, en línea con los requerimientos de seguridad proporcionados por Azure.

El proceso de inicio de sesión se lleva a cabo mediante la generación y validación de tokens. Los usuarios deben contar con los tokens correspondientes para poder acceder a la API. Estos tokens actúan como credenciales de autenticación, garantizando la legitimidad del usuario y autorizando su acceso a los recursos protegidos.

Además de los tokens, se utilizan cookies para facilitar el inicio de sesión y mantener la sesión activa. Las cookies son pequeños archivos de texto almacenados en el navegador del usuario. Al iniciar sesión, se genera una cookie que contiene información relevante, como un identificador único de sesión.

Durante las solicitudes subsiguientes a la API, el navegador adjunta automáticamente la cookie pertinente, permitiendo así que la API identifique la sesión del usuario y valide su acceso en base a la información almacenada en la cookie.

En cuanto a la seguridad, el proyecto ha aprovechado los servicios proporcionados por Azure, en particular Azure Active Directory (Azure AD). Esta plataforma de administración de identidades y accesos se encarga de la emisión y validación de los tokens de seguridad necesarios. De esta manera, se asegura que solo los usuarios autenticados y autorizados puedan acceder a los recursos protegidos por la API.

En relación a la duración de la sesión, se ha implementado la funcionalidad de expiración de cookies. Después de transcurrir un período de tiempo determinado, las cookies caducan y, como resultado, el usuario es deslogueado automáticamente del sistema. Esta medida de seguridad se implementa para prevenir accesos no autorizados a la cuenta del usuario en caso de que olvide cerrar sesión o abandone el dispositivo sin protección.

En conclusión, el proyecto ha integrado con éxito el uso de tokens y cookies para el inicio de sesión y la obtención de información de la API. La seguridad proporcionada por Azure, a través de Azure AD, garantiza la autenticación y autorización adecuadas de los usuarios. Además, la expiración de cookies brinda una capa adicional de seguridad al desloguear al usuario después de un cierto período de tiempo. Estas implementaciones aseguran la protección de los recursos y la privacidad de los usuarios en el marco de este proyecto.

1. **Depuración y solución de problemas.**

Para identificar y comprender adecuadamente los problemas específicos relacionados con la arquitectura de mi proyecto, se recomienda seguir estos pasos:

a) Reproducción del problema: Intentar recrear el problema de manera sistemática para comprender su origen y los pasos necesarios para replicarlo de manera consistente.

b) Análisis de registros y mensajes de error: Examinar los registros de la aplicación y los mensajes de error para obtener información relevante sobre el problema en cuestión. Esto permitirá comprender dónde y por qué se está produciendo el error en el contexto de la API ASP.NET Core, la base de datos SQL Server y la interacción con el frontend en React.

Solución de problemas:

Una vez que se haya identificado un problema específico, es fundamental seguir estos pasos para su resolución:

a) Verificación de la configuración: Asegurarse de que todas las configuraciones necesarias para la API ASP.NET Core y la conexión a la base de datos SQL Server estén correctamente establecidas. Esto incluye comprobar las cadenas de conexión, la configuración de red y cualquier otro aspecto relevante.

b) Examen del código: Analizar minuciosamente el código relacionado con el problema específico en la API ASP.NET Core y en el frontend de React. Revisar si existen errores de sintaxis, llamadas a funciones incorrectas o lógica inadecuada que puedan estar causando el problema.

c) Depuración: Utilizar herramientas de depuración disponibles en el entorno de desarrollo de la API ASP.NET Core y React para examinar el estado de las variables, rastrear la ejecución del código y detectar posibles problemas específicos en la interacción entre la API, la base de datos y el frontend.

d) Pruebas unitarias: Implementar pruebas unitarias específicas para las diferentes partes de la API ASP.NET Core y el frontend de React. Estas pruebas unitarias deben abarcar escenarios relevantes para identificar rápidamente problemas y errores en segmentos específicos de código.

Posibles conflictos con otros sistemas o componentes:

Durante el desarrollo de mi proyecto, es importante tener en cuenta posibles conflictos con otros sistemas o componentes específicos en el entorno de Azure. Aquí hay algunas recomendaciones para evitar o resolver estos conflictos en el contexto de mi proyecto:

a) Verificación de la configuración de Azure: Asegurarme de que la configuración de Azure, como la configuración del servidor web, las reglas de enrutamiento y la asignación de recursos, esté correctamente establecida y sea compatible con la arquitectura de mi proyecto.

b) Monitoreo de los recursos de Azure: Utilizar herramientas de monitoreo de Azure para detectar posibles problemas de rendimiento, latencia o escalabilidad en la interacción entre la API ASP.NET Core, la base de datos SQL Server y el frontend de React alojados en Azure.

c) Actualización de las versiones y compatibilidad: Mantener actualizadas las versiones de los componentes utilizados en Azure, como los servicios de base de datos SQL Server, las bibliotecas y las herramientas de desarrollo, para garantizar la compatibilidad y evitar conflictos conocidos.

1. **Glosario de términos.**

**Base de datos:** Programa capaz de almacenar gran cantidad de datos, relacionados y estructurados, que pueden ser consultados rápidamente de acuerdo con las características selectivas que se deseen.

**Azure Portal:** Es un servicio de hosting

**API:** significa “interfaz de programación de aplicaciones”. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones.

**Cookies:** es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del navegador.

1. **Referencias y recursos adicionales.**

<https://youtu.be/MLFU3CkOd1o>

<https://youtu.be/HEFXebTGiMA>

<https://youtu.be/XVcyVnNWUwo>

<https://youtu.be/rLoWMU4L_qE>

<https://codigofacilito.com/cursos/react-api-store>

<https://codigofacilito.com/cursos/javascript-profesional>

1. **Herramientas de implementación.**

Visual Studio.

Visual Studio Code.

SQL Server

GIT HUB

Azure Portal

Azure Blob Storage

Azure DevOps

1. **Bibliografía.**

<https://tailwindui.com/documentation>

<https://tailwindcss.com/docs/installation>

<https://legacy.reactjs.org/docs/getting-started.html>