Prueba técnica – Inalambria

Jose Daverson Parra Barrero

1. Teniendo en cuenta la necesidad del cliente y el crecimiento exponencial se diseña una arquitectura de componentes la cual permita una rápida implementación y desarrollo teniendo en cuenta las consideraciones descritas dentro de la necesidad, con una metodología ágil en afinación con DevOps.

Para ello se tuvo en cuenta un sistema que soporte una alta disponibilidad de sus servicios utilizando Docker + kubernetes transversalmente, administrados con AZURE y monitoreados con ELK.

Para el desarrollo de API, se desarrollará bajo la arquitectura de microservicios bajo los principios SOLID para su codificación, así como de la implementación de patrones como el Repository para el acceso a la capa de persistencia teniendo en cuenta las diferentes fuentes y tipos de datos que se almacenarán.

Para el desarrollo front se desarrollará bajo la arquitectura hexagonal, separando los artefactos de tal manera que su desarrollo y mantenimiento sea el más optimo en síntesis con las metodologías que se adoptaran para su despliegue y gestión.

1. En cuanto a seguridad, la implementación de autenticación y autorización se realizará con JWT para las API, orquestación del API Gateway con NGINX, monitorización a través de ELK.
2. El proyecto deberá contar con 4 ambientes que permitan soportar un despliegue continuo:
   1. TST – Ambiente de desarrollo
   2. QA – Ambiente de pruebas de calidad
   3. UAT – Ambiente pre productivo
   4. PROD – Ambiente productivo

Los despliegues se realizarán por medio de pipelines que permitan integrar el ciclo de desarrollo automatizando los despliegues con integración de AZURE.

Las condiciones de maquina mínimas se darán en base a las pruebas de carga y estrés a través del monitoreo dado por ELK, con lo cual se dará una escalabilidad de la infraestructura basada en los datos y el crecimiento estimado.

1. Se entrega desarrollo adjunto, para el cual y por tiempos de la prueba se opta por realizar así:
   1. El Back-end se desarrolla con una arquitectura orientada a microservicios en ASP.NET (.NET Framework) c# en su versión de .NET Framework 4.7.2, teniendo en cuenta que el soporte previsto supera los 5 años y no cuenta con registros de vulnerabilidades en CVE (las 13 detectadas cuentan con parches de seguridad que las mitigaron como se puede ver en el MSRC), es un lenguaje con un amplio soporte y comunidad, lo cual lo hace ampliamente competitivo en el mercado.

A nivel de librerías CORE se pueden tienen las siguientes capas:

* + 1. Lógica de negocio
    2. Entidades
    3. Globalización
    4. Repositorio
    5. Unidad (Abstracción del patrón Repository con EF)
    6. Utilidades (Extensiones o métodos transversales a la solución)

Las API correspondientemente infieren con una única entidad de negocio generando una respuesta Json con una estructura definida como estándar para su implementación a nivel de front conteniendo toda la información necesaria sobre la resolución de la petición.

* 1. El front-end se desarrolla con una arquitectura hexagonal, con el framework VueJS 2.x, teniendo en cuenta su ecosistema, documentación y curva de aprendizaje. El proyecto se encuentra VUE CLI, vue router, para el enturado, vuex para el manejo de estados, vuelidate para la validación del formulario de creación de usuarios, axios para las solicitudes http, con framework vuetify para la interfaz, se separan las vistas mostrando únicamente las secciones a las cuales tiene permisos el rol del usuario ingresado, al igual que las acciones sobre esta, se tiene separada la capa de servicios que es aquella que realiza las peticiones HTTP a través de axios y cada servicio apunta a API específica, retornando una promesa, la cual es medidada desde el store, las vistas y componentes únicamente llaman acciones del store y este es el que se encarga de ejecutar el respectivo servicio y resolver las promesas y muta el store para que las vistas de forma reactiva se rendericen.
  2. Se dispone del código de los proyectos:
     + - Front - ~\prueba-tecnica\front-end
       - Back - ~\prueba-tecnica\prueba-tecnica

como de los publicados:

* + - * Front - ~\prueba-tecnica\publicados\front
      * Back - ~\prueba-tecnica\publicados\api
      * BD - ~\prueba-tecnica\bd

Si se quiere realizar la publicación para revisión se debe tener en cuenta las siguientes consideraciones a nivel de base de datos y configuración:

* + - * Al ejecutar el archivo de bd se creará la BD pruebadb y con ella por defecto un usuario de rol admin para el aplicativo
        1. usuario: admin
        2. contraseña: 123456
      * A nivel de BD, se requiere que exista un login con autenticación de SQL Server
        1. Login: pruebauser
        2. Contraseña: clave123

con una asignación de usuario a la base pruebadb y rol db\_owner.

* + - * En el web.config del publicado Back, se deberá actualizar el data source de las connectionString apuntando al server donde se encuentra alojado el motor.
      * La publicación del back se recomienda hacerla en el puerto 59538, la cual se encuentra configurada dentro del archivo .env.production del proyecto Front, si se requiere cambiar el puerto, se puede realizar en el archivo mencionado y generar nuevamente la publicación (npm run build).
      * El server de despliegue debe tener instalado .NET Framework 4.7.2 Runtime.
      * Teniendo en cuenta que la publicación del back se debe realizar en IIS (versión 10 o posterior y el Pool de aplicaciones con versión de .NET CLR v4.0) se puede desplegar el FRONT también en el IIS.