Para esta solución se piensa en una aplicación monolítica con una arquitectura basada en capas utilizando dos componentes para la interacción con la aplicación, el frontend con Angular y el backend hecho con .NET Core, la comunicación entre estos componentes se piensa realizar a través de API Rest con una API Management como intermediario de la conexión con tal de garantizar seguridad de la información de accesos, finalmente el acceso a la base de datos se piensa proteger también utilizando los servicios de Azure Key Vault para enmascarar las cadenas de conexión de la base de datos, finalmente la base de datos en Azure Cosmos DB (API SQL).

Como metodología de desarrollo utilizaremos una metodología ágil en base a SCRUM con enfoque iterativo e incremental con los usuales pasos en cascada para la planeación, definición de requerimientos o historias de usuario, adicionalmente y debido a la naturaleza del enfoque en servicios basados en la nube de Azure, es necesario utilizar una metodología personalizada que involucre este paso debido a que usualmente no se realiza un paso donde se tenga en cuenta la implementación de estos servicios, posteriormente el desarrollo de cada uno de los componentes que conforman la aplicación web, ósea el desarrollo del frontend en Angular y el backend en .NET Core, estos preferiblemente deban hacerse simultáneamente para evitar implementación extensas debido a cambios que puedan surgir en el desarrollo de cada uno, posteriormente la realización de pruebas, implementación y despliegue, y finalmente monitoreo y mantenimiento. La realización de pruebas se está pensado realizar simultáneamente también con el desarrollo con el uso de pruebas unitarias para poder realizar además la implementación y despliegue con herramientas de testing que nos permitan poder tener un mayor control a lo que se despliega con herramientas como Sonar que nos permitirán también tener un control de código limpio y donde podemos configurarla para automatizar tareas, el paso de implementación y despliegue se realizar con las herramientas de Azure para el control de versionamiento, releases y pipelines.

Con esta metodología de desarrollo podemos tener una mayor colaboración y adaptación continua a cambios, es eficiente incorporar integración y despliegue continuos (CI/CD) para la automatización de pruebas y una mayor facilidad del despliegue que permiten mitigar errores y nos permiten entregas mas rápidas y seguras en los distintos ambientes que se dispongan en este proyecto, podemos utilizar control de versiones herramienta esencial para el manejo del código y que permite la trazabilidad de cambios y el desarrollo en paralelo, el code review o revisión de código también se vuelve esencial para la mitigar errores antes de que se desplieguen a producción y para la sincronización del equipo de trabajo, refactorización continua para la mantenibilidad y escalabilidad no solo del código sino de la aplicación, podemos reducir errores y deuda técnica de esta forma y finalmente documentación en constante actualización con el desarrollo y los cambios efectuados por distintas razones durante el desarrollo.

La aplicación se desarrollará como una SaaS o software como un servicio de están forma evitamos más riesgos y reducimos el alcance del proyecto, por lo que se hace primordial contar con al menos 3 ambientes de producción para garantizar la calidad y las entregas y mejoras continuas, un ambiente de desarrollo, un ambiente de pruebas y uno de producción, para tecnologías colaborativas se utilizar los servicios de Microsoft por su integración y comodidad con lo requerido para este proyecto, Teams para la comunicación entre el equipo y Azure DevOps para la definición y asignación de tareas.

Posiblemente durante el desarrollo surjan cambios por lo que es necesario establecer comunicación y participación activa con y para el cliente durante todo el ciclo de vida del software, se pueden presentar problemas de desempeño con el equipo de trabajo por lo que es necesario conta con el personal adecuado, y capacitar al personal cuando sea necesario, además se pueden presentar problemas en la infraestructura y ya que la arquitectura esta basada en gran parte en servicios en la nube será necesario selecciona cuidadosamente los servicios requeridos e implementar estrategias que permitan realizar cambios sobre la marcha pero lo primordial es realizar un buen análisis de estos servicios, es posible también que se presenten problemas de rendimiento de la aplicación, por lo que será necesario contar con pruebas de carga y rendimiento durante la fase de pruebas y de implementación y despliegue, también contar con código optimo y organizado, pueden surgir también retrasos en el desarrollo, por lo que es necesario durante la fase de planeación realizar metas u objetivos realistas además con esta metodología de desarrollo podemos realizar entregas sobre el desarrollo para evitar cambios posteriores al desarrollo que obliguen al equipo a retomarlo, es necesario resaltar de nuevo la comunicación con el cliente o involucrados para que se puedan tomar decisiones a situaciones de mayor riesgo.

Angular (Frontend): Se utilizará Angular debido a su amplio conjunto de herramientas para el desarrollo, su estructura de componentes y su capacidad modular, además de utilizar TypeScript para facilitar código mantenible y escalable, además cuenta con una amplia y muy detallada documentación oficial y no oficial a través de sus múltiples colaboradores, también posee soporte para el desarrollo de aplicaciones moviles.

.NET Core (Backend): Es el framework ideal para crear aplicaciones que utilicen servicios en la nube de Microsoft Azure, se complementa muy bien con estos además de ofrecer implementación nativa con estos, ofrece además la capacidad de ser multiplataforma en caso de ser necesario alojarlo en otros entornos, su rendimiento es considerablemente optimo, es altamente configurable, posee soporte para implementación API Rest, además de ofrecer múltiples herramientas de seguridad, también es optimo para el desarrollo de arquitecturas basadas en microservicios por lo que ofrece también escalabilidad, posee también herramientas ideales para el desarrollo de pruebas, posse también soporte para señales en tiempo real.

Azure API Management: Es altamente personalizable a las necesidades de un proyecto, es integrable con otras herramientas de servicios en la nube de azure ya que corresponde al mismo grupo, ofrece un alto grado de seguridad de la comunicación entre componentes además de ofrecer herramientas de monitoreo y análisis.

Azure Key Vault: Otro servicio del conjunto de servicios en la nube de Azure que nos permite encriptar o enmascarar cadenas de conexión u otras llaves secretas o de acceso, es fácilmente integrable con otros servicios de Azure y ofrece alta escalabilidad.

Azure Cosmos DB (API SQL): Es ideal utilizar Cosmos DB de Azure debido a que se integra eficientemente con otro servicios en la nube de Azure, además permite escalabilidad, confiabilidad y rendimientos necesarios para la carga de este proyecto así como también permite alta configurabilidad que se ajuste no solo a las necesidades del proyecto sino además a las necesidad temporales gracias a su capacidad de actuar bajo demanda, por lo que su coste es bastante flexible, cuenta también con otras herramientas propias de los servicios en la nube de Azure como análisis en tiempo real y alta disponibilidad