Tabla de Contenidos

1. INTRODUCCIÓN

Una arquitectura de referencia es una descripción técnica que describe las consideraciones de diseño relevantes para el tipo de solución que describe, prescribiendo aspectos tanto funcionales como no funcionales incluyendo diseños específicos, guías y recomendaciones de arquitectura, operaciones, despliegue y seguridad, asegurando que el diseño se adhiera a prácticas reconocidas y probadas por la industria, permitiendo que la solución se pueda entregar y operar de manera óptima.

Propósito

Este documento describe la arquitectura de referencia para las aplicaciones Web que se desarrollarán en el Ministerio de Agricultura y Desarrollo Rural en el marco del centro de excelencia para productores 360. El centro de excelencia nace para articular la implementación de las necesidades identificadas en el ejercicio de arquitectura empresarial que se desarrolló en el ministerio durante 2014, donde surgió Productores 360 como una iniciativa fundamental que le dé al ministerio las herramientas que necesita para cumplir sus objetivos.
Productores 360 abarca aspectos de tecnología, procesos y con otras entidades. De los aspectos tecnológicos se identificaron un subconjunto de aplicaciones que se categorizan como aplicaciones Web y se refiere esencialmente a aplicaciones que se utilizarán a través de un navegador (*Browser*) y mediante una conexión a Internet. Este componente describe la arquitectura de referencia para esta clase de aplicaciones.

Audiencia

Este documento está dirigido a:

- Arquitectos empresariales
- Arquitectos de soluciones
- Arquitectos de aplicaciones
- Desarrolladores
-

Supuestos

Se asume que el lector conoce la plataforma de desarrollo de aplicaciones y comprende las generalidades de Azure.

2. Contexto de negocio

Como resultado del ejercicio de arquitectura empresarial se identificaron una serie de necesidades junto con las recomendaciones tecnológicas para resolverlas. En particular, Productores 360 tiene que ver con la necesidad de centralizar la información de los productores, permitirles a esta directa o indirectamente registrarse y actualizar sus datos, y permitirle al ministerio y demás interesados consumir esta información para tomar decisiones y utilizarla para brindarles beneficios a los grupos o zonas que elijan.
Los escenarios identificados hasta ahora en la arquitectura empresarial y por el centro de excelencia son:

Usuario	Ubicación	Uso
Productores, público en general	Cualquier ubicación de internet, zonas de acceso limitado a internet	Llenar formularios de registro Actualizar sus datos de registro Consultar información estadística Recibir notificaciones vía SMS u otro mecanismo
Funcionarios de entidades territoriales, entidades adscritas y otras entidades del sector	Oficinas de la entidad, a través de conexiones a internet de banda ancha	Registrar productores individual y masivamente Actualizar la información de los productores individual y masivamente Consultar información estadística Consultar reportes específicos a sus intereses Participar en flujos de trabajo en roles de aprobar / rechazar y proveer información
Funcionarios del ministerio	Oficinas del ministerio	Todas las anteriores Diseñar los flujos de trabajo Parametrizar el sistema

La arquitectura empresarial recomienda utilizar Dynamics CRM/xRM como el eje central para guardar esta información, referencia *capítulo 5.3 Materialización de la visión* del documento *Opciones y recomendaciones de arquitectura*.
Esta recomendación trae una cantidad importante de beneficios, entre ellos:

- Flexibilidad en el modelo de dominio: En Dynamics CRM las entidades se crean y modifican vía interfaz gráfica, sin producir indisponibilidad de la aplicación, y en la mayoría de los casos sin necesidad de cambios en código .NET
- Muchos casos de uso ya resueltos: Dynamics CRM brinda como parte del producto muchos casos de uso que en el caso de un desarrollo a la medida requerirían todo el esfuerzo de desarrollo. Por ejemplo todas las pantallas para crear y mantener datos de referencia hacen parte del producto así como diseñadores de formularios y diseñadores de flujos de trabajo (*Workflows*).
- Muchos casos de uso se logran vía configuración: La mayoría de flujos de trabajo, definición de entidades, restricciones en valores entre otras cosas se logran vía configuración en vez de mediante programación.
- Aspectos no funcionales ya resueltos: Aspectos como auditoria, modelo de seguridad, accesibilidad y escalabilidad están ya disponibles en el producto.

Estos beneficios, más que beneficios pueden considerarse requerimientos para el ministerio pues sus definiciones cambian frecuentemente. Si el sistema que soporta sus funciones esenciales no es capaz de cambiar al tiempo entonces el sistema no será adecuado para las necesidades del ministerio.
Dynamics CRM es ideal para cobijar las necesidades del ministerio al interior del ministerio, sin embargo fuera del ministerio la solución óptima es una aplicación Web.
Si bien Dynamics CRM es también una aplicación Web no será óptimo para los usuarios fuera del ministerio por las razones descritas a continuación:

- El universo de estos usuarios comprende funcionarios de entidades territoriales como alcaldías y comisarias, otras entidades adscritas, asociaciones de productores, los mismos productores e incluso cualquier ciudadano. Dynamics CRM se licencia por usuario lo que dado el universo de usuarios descrito hace inviable brindarles acceso directo a Dynamics CRM.
- Según ha comentado el ministerio de experiencias anteriores con aplicaciones Web, una importante mayoría de sus usuarios tiene restricciones de conectividad relacionadas con muy difícil acceso a internet (deben desplazarse trayectos significativos para llegar a un sitio con conectividad) y con anchos de banda muy limitado (comparable con internet telefónico 56kbps o satelital 128kbps). Las interfaces de usuario de Dynamics CRM son de tipo **RIA** - *Rich Internet Application* brindándole al usuario una experiencia similar a la de una aplicación de escritorio. Esta clase de experiencia funciona muy bien con anchos de banda

adecuados, sin embargo no es recomendable fuera de una intranet o una conexión de banda ancha (>2mbps).

Las aplicaciones Web brindarán entonces las interfaces de usuario para usuarios finales fuera del ministerio para los casos de uso que así lo requieran. Cabe aclarar que Dynamics CRM seguirá siendo el repositorio y motor de lógica de negocios para estos casos de uso, solamente que no brindará directamente la interfaz de usuario para estos.

3. Azure

Azure es una plataforma de servicios en la nube, de escala Internet.

De los servicios que ofrece Azure esta arquitectura de referencia utilizará los siguientes:

- App Service
- SQL Database
- Cache
- Storage
- Active Directory
- Virtual Network
- Traffic Manager
- CDN

App Service

Es un servicio de nube en plataforma como servicio que permite crear sitios web y móviles para cualquier plataforma y dispositivo. App Service agrupa varios servicios, Web Apps, Mobile Apps, API Apps y Logic Apps. Ref: https://azure.microsoft.com/en-us/documentation/articles/app-service-changes-existing-services/?clcid=0x409

En este servicio se alojarán los sitios web como tal pero aprovechando todas las ventajas de plataforma como servicio, entre ellas escalabilidad, integración continua y múltiples 'cajones' de despliegue.
Ref: https://azure.microsoft.com/en-us/documentation/articles/app-service-web-overview/

SQL Database

Básicamente SQL en la nube como plataforma como servicio. Aunque existen varias limitaciones respecto a la versión para servidor de SQL ninguna de estas limitaciones

afectan negativamente el diseño de esta solución.
El repositorio preferido para la información de productores 360 será Dynamics CRM, sin embargo cuando se identifiquen datos estructurados que no encajan en Dynamics CRM entonces se guardarán en SQL Azure.
Ref: https://azure.microsoft.com/en-us/services/sql-database/

Cache

Servicio de cache en la nube como plataforma como servicio.
Cuando se identifiquen datos cuyo tiempo de recuperación esté afectando el tiempo de respuesta de las páginas, o datos que se consulten frecuentemente a base de datos o a Dynamics CRM afectando su propio desempeño se evaluará llevarlos a Azure Cache.
El criterio de evaluación en este caso es que tan frecuentemente cambian.
Ref: https://azure.microsoft.com/en-us/services/cache/

Storage

Dentro de los servicios de almacenamiento de Azure se encuentran Blobs, Tables y Queues. En este diseño se hará uso de todas estas estructuras de almacenamiento pues son parte fundamental en la materialización de los patrones de diseño que la componen.
Los Blobs se utilizan para guardar data no estructurada, por ejemplo imágenes.
Tables son datos estructurados pero no relacionales, de forma masivamente escalable sacrificando las búsquedas relacionales.
Queues son colas, utilizadas para almacenamiento confiable y mensajería entre capas o roles.
Ref: https://azure.microsoft.com/en-us/services/storage/

Active Directory

Repositorio de identidad en la nube, servicios de autenticación y servicios de identidad federada.
Ref: https://azure.microsoft.com/en-us/services/active-directory/

Virtual Network

Servicios de red y VPN. En la medida que el diseño físico de productores 360 y otras soluciones del ministerio compartan datos desde y hacia el centro de datos propio del ministerio tiene sentido tener conectividad directa hacia Azure via VPN.
Ref: https://azure.microsoft.com/en-us/services/virtual-network/

Traffic Manager

Enrutador a nivel de internet para brindar mejor desempeño y disponibilidad. La arquitectura física de la solución contempla alta disponibilidad con redundancia geográfica y mediante este servicio se llevará el tráfico al centro de datos más cercano al usuario.
Ref: https://azure.microsoft.com/en-us/services/traffic-manager/

CDN

Servicio de distribución de contenido. Con este servicio se optimiza la entrega de contenido estático de gran volumen como videos, audio e imágenes de gran tamaño.
Ref: https://azure.microsoft.com/en-us/services/cdn/
Ref: https://azure.microsoft.com/en-us/services/service-bus/

Application Insights

Monitoreo y alertas de disponibilidad y desempeño de aplicaciones web y móviles.
Ref: https://azure.microsoft.com/en-us/services/application-insights/

Otros Servicios

Existen otros servicios que pueden ser de utilidad en el ministerio según los requerimientos de integración con otras entidades. Estos servicios son Azure Biztalk Service, Azure Service Bus y Azure API Management.

Movilidad hacia otros proveedores o al centro de datos

Este diseño asume el uso de Azure. De los servicios enumerados anteriormente existen equivalentes o similares tanto en otros proveedores de nube como en una instalación tradicional en centro de datos. Debe mencionarse sin embargo que un movimiento de esta clase pondrá en entredicho los aspectos no funcionales que promueve este diseño como disponibilidad y escalabilidad, posiblemente limitará las habilidades de despliegue continuo y necesariamente implicará cambios menores en el código.

4. Principios

Este diseño se rige por los principios enumerados en los entregables “ALM” y “Recomendaciones de codificación”, así como los enumerados en la guía de arquitectura de aplicaciones .NET, ref: https://msdn.microsoft.com/en-us/library/ee658124.aspx
Además de estos principios generales a continuación se enumeran los principios particulares para este tipo de aplicación.

Principios de aplicación

Asincronismo

Todas las llamadas que cruzan una capa lógica o física deben ser de tipo asincrónico. Del lado servidor el uso de llamadas de tipo asincrónico permite hacer uso óptimo de los recursos computacionales al no bloquear el procesamiento mientras se espera por la respuesta de otra operación.
Del lado cliente la llamada asíncrona permite al sistema operativo reaccionar a otras acciones del usuario y potencialmente retroalimentar al usuario del avance de su solicitud.
Para esta clase de aplicaciones debe aprovecharse las características que se han estado introduciendo desde .NET 4.5 con async/await y el soporte para operaciones async en la inmensa mayoría de librerías .NET

Manejo de fallas transitorias

El código debe esperar que cualquier llamado hacia otra capa física falle de forma transitoria y debe estar preparado para recuperarse de tales fallas.

Por ejemplo, cuando el código llame una rutina cuyo fin es guardar algo en SQL Azure o consultar algo en Dynamics CRM, si esta operación falla, el código no debe darse por vencido inmediatamente pues puede tratarse de una falla transitoria que al volverlo a intentar posiblemente funcione.

En el caso de Entity Framework esta funcionalidad hace ya parte del Framework. En otros casos se recomienda utilizar el bloque de código de manejo de fallas transitorias disponible en:
https://msdn.microsoft.com/en-us/library/dn440719(v=pandp.60).aspx

Instrumentación

Todo llamado que cruce una capa lógica o física debe cronometrarse y registrarse. En esta clase de aplicaciones el tiempo de respuesta hace la diferencia, por eso es necesario contar con datos precisos que permitan detectar rápidamente cuellos de botella.
Para cronometrar el llamado debe utilizarse un objeto Stopwatch en vez de Datetime. Para registrar el tiempo utilizar tanto debug.Print como la facilidad de *Application Insights* para recibir registros.

Principios de arquitectura y diseño

Arquitectura por capas

El estilo de arquitectura por capas separa la aplicación en diferentes capas de responsabilidad donde las capas inferiores desconocen las capas que las invocan. De esta forma una capa puede invocarse por cualquier otra capa y reutilizarse para distintos consumidores. Esta separación de responsabilidades promueve la mantenibilidad. En este diseño se utilizará un modelo clásico de capa de datos, capa de servicio y capa de presentación. Esta separación lógica no siempre es física; hacerlo a nivel físico promueve la escalabilidad pero en el caso de aplicaciones Web frecuentemente agrega complejidad que hace cuestionable su conveniencia. Para el caso de las aplicaciones Web de Productores 360 no se prevé conveniente introducir separación física de capas y esto se refiere concretamente a no separar la aplicación de presentación de la aplicación de servicios.

Separación de aspectos

Es la práctica de diseño mediante la cual cada aspecto está plenamente diferenciado y hay muy poco solapamiento entre funcionalidades. Esto se logra mediante modularidad, encapsulación y el principio de única responsabilidad. Con esta práctica se promueve la mantenibilidad y flexibilidad de la aplicación.

Inversión de control

Es una forma de reducir el acoplamiento al no requerir objetos concretos en cada llamada sino interfaces de cómo es uno de esos objetos. Esta técnica además de reducir el acoplamiento permite fácilmente intercambiar componentes y aumenta la factibilidad de introducir pruebas unitarias. En la implementación se utilizará el inyector de dependencias provisto por el Framework que en tiempo de ejecución brindará los objetos concretos que se requieran.

MVC – Modelo Vista Controlador

Este patrón divide la aplicación en tres áreas de responsabilidad:

- El modelo, dueño del modelo de negocio incluyendo sus datos y lógica
- La vista, que genera la interfaz gráfica y captura las entradas del usuario
- El controlador, acepta solicitudes del usuario, traduce las entradas del usuario a operaciones en el modelo y entrega datos a la vista para crear la interfaz de usuario

El modelo típicamente necesita una base de datos u otra clase de sistema de soporte como Dynamics CRM, pero no debe ocuparse de los detalles de cómo comunicarse con este sistema, para esto se introduce una cuarta área de responsabilidad. El responsable de interactuar con el sistema de soporte es un “repositorio”.

5. Atributos de calidad

Escalabilidad y elasticidad

Las aplicaciones Web de Productores 360 deben ser escalables y elásticas pues el volumen de usuarios y su patrón de uso no se controla y no se conoce del todo. Las aplicaciones Web son relativamente fáciles de escalar sobretodo en Azure siempre y cuando se sigan los patrones de arquitectura adecuados.

La escalabilidad consiste en agregar o quitar instancias de cómputo al servicio para suplir la demanda de usuarios manteniendo el tiempo de respuesta propuesto.

Quiere decir que la aplicación debe ser capaz de atender correctamente una petición desde cualquier instancia y para esto la aplicación debe ser "*stateless*" o "sin estado". Quiere decir que las instancias no mantienen ningún dato acerca del usuario, lo que estaba haciendo antes, o cualquier concepto similar a sesión. En vez de esto toda la información que el cliente necesita para ser atendido la envía en cada petición, o, envía un identificador con el que la instancia puede consultarla en un repositorio. Estas dos técnicas no son excluyentes y en Productores 360 se utilizarán las dos aplicando como criterio los demás atributos de calidad.

Otra forma de aumentar la escalabilidad consiste en darle manejo asíncrono a las operaciones de larga duración. Este asíncrono no se refiere al del punto 0 que tiene que ver con codificación sino a la interacción con el usuario. Si el usuario no necesita una respuesta inmediata o de antemano se sabe que la operación va a tardar más de un par de segundos en completarse es mejor reconsiderar el caso de uso para que el usuario sea notificado del resultado cuando esté disponible.

Respecto a la operación de agregar y quitar instancias esto se hace vía configuración en Azure, de forma automática en base a mediciones como tiempo de respuesta o carga de procesamiento, o en base a fechas u horas específicas.

Disponibilidad y confiabilidad

Las aplicaciones Web de productores 360 estarán expuestas al público y se espera una disponibilidad permanente.
Aunque Azure está construido precisamente para brindar disponibilidad y confiabilidad

su SLA no es del 100%, y aun para aplicar al SLA de 99.9% el diseño físico debe contemplar al menos dos instancias corriendo en cualquier momento.
Aun así, sigue existiendo el riesgo de perder por completo el centro de datos donde se encuentran las dos instancias por lo que el diseño físico debe además contemplar redundancia geográfica.
La configuración óptima para este atributo de calidad se logra desplegando la solución en dos centros de datos geográficamente separados. Para el caso de Colombia al momento de elaborar este diseño los centros de datos recomendados son este y oeste de Estados Unidos.

Seguridad

Las aplicaciones Web disponibles en Internet están expuestas a toda clase de amenazas independientemente de su propósito y tecnología pues hoy en día en Internet abundan ataques automatizados que rastrean sitio por sitio en busca de posibles vulnerabilidades. Esto sumado a los ataques específicos contra el gobierno Colombiano o el ministerio convierten la seguridad en una prioridad permanente.
Al hospedar los sitios en una solución de nube en plataforma como servicio muchos aspectos se delegan en el proveedor de servicios; por ejemplo todo lo relacionado con seguridad física del centro de datos y parchado de los sistemas operativos que soportan la plataforma son temas que se delegan por completo en Azure.

Seguridad de aplicación

A nivel de aplicación debe prestarse especial atención a los aspectos identificados por OWASP, http://www.owasp.org/,
ref: https://asafaweb.com/OWASP%20Top%2010%20for%20.NET%20developers.pdf

A continuación se describe cada uno de los puntos en el top 10 de OWASP

Seguridad en el transporte

Las aplicaciones Web del ministerio solo estarán disponibles bajo SSL/TLS. Esta tecnología previene la intercepción, suplantación y alteración de la información.

Prevención de ataques por inyección de comandos

En un ataque por inyección el atacante engaña al sitio web con una petición que parece legítima y en realidad inyecta comandos tipo SQL, LDAP, o del sistema operativo a la plataforma conduciendo a la divulgación de información privilegiada, alteración del sitio web o suspensión del servicio. El Framework por defecto previene muchos de estos ataques bloqueando automáticamente peticiones que se conoce son maliciosas.

Aunque esta tecnología en general funciona no por eso debe descuidarse en la codificación prácticas recomendadas como siempre utilizar consultas parametrizadas.

Prevención de ataques XSS

XSS se refiere a correr scripts de un sitio en otro, secuestrando la sesión del usuario o haciéndole creer que está introduciendo datos en un sitio confiable cuando en realidad está un sitio malicioso. El Framework incorpora tecnologías para evitar estos ataques, tecnologías que por defecto están activadas y no se deben desactivar. Además, en la codificación, deben utilizarse explícitamente los mecanismos de codificación que brinda el Framework para prevenir estos ataques, en particular cualquier salida de texto debe pasar por Server.HtmlEncode antes de entregarse a la vista y cualquier entrada de texto debe pasar por AntiXssEncoder.HtmlEncode antes de procesarse.

Manejo de autenticación y sesión

Los mecanismos de autenticación y manejo de sesión que brinda por defecto el Framework para páginas Web son adecuados desde la perspectiva de seguridad, el Framework sin embargo no incluye por defecto mecanismos de seguridad para los servicios REST API. En este punto para el sitio Web debe verificarse que el cookie de sesión está marcado como cookie de sesión, solamente válido para el dominio del sitio web del ministerio y que expire. En REST API para las operaciones que son dependientes del usuario debe utilizarse autenticación basada en tokens JWT.

Prevención de ataques de falsificación de sitios

Estos ataques se denominan *CSRF (Cross-Site Request Forgery)* y hacen que el navegador de la víctima envíe sus peticiones, incluyendo el cookie de sesión, a un sitio malintencionado sin que el usuario se dé cuenta. El Framework incluye tecnologías para prevenir estos ataques, tecnologías que deben explícitamente utilizarse durante la implementación. En particular tanto en formularios web como en POST para REST API debe incluirse un "*Anti forgery token*", provisto por el Framework y que previene esta clase de ataques.

Vulnerabilidades por configuración

Un vector frecuente de ataque tiene que ver con la configuración. El principal problema con estas vulnerabilidades es que su prevención es un ejercicio permanente. En la

práctica esto implica estar atento a los boletines de seguridad y de los proveedores de todos los componentes utilizados en la solución. Si bien todo lo relacionado con la plataforma se delega en Azure aspectos de configuración como la versión de jquery utilizada o aspectos particulares de la implementación donde se descubran vulnerabilidades deben ser mitigados por el ministerio aplicando las actualizaciones o recomendaciones del caso.

Referencias directas inseguras

Como la aplicación carece de estado ("*Stateless*") es frecuente incluir referencias directas en las peticiones, ej: madr.org/productores/8798223 . Esta clase de referencias facilita la construcción de la aplicación y el reúso de direcciones de internet por parte de los usuarios, sin embargo un usuario malintencionado podría manipular esta referencia para lograr acceso a datos de otros usuarios o datos a los que no debería tener acceso. El ejemplo muestra la URL, sin embargo esta vulnerabilidad se extiende a cualquier dato que transite entre el usuario y el servidor pues es trivial para el usuario inspeccionar y manipular estos datos. En particular incluye HTTP POST, peticiones JAVASCRIPT, QUERYSTRING, COOKIES, HTTP HEADERS.
La mitigación de estas vulnerabilidades no consiste en eliminar estas referencias, consiste en asegurar que en la codificación las rutinas que atienden estas peticiones verifiquen siempre la identidad del usuario y a partir de ahí filtren la información a la cual este tiene acceso bien sea por su rol o porque solo tiene acceso a su propia información.

Almacenamiento criptográfico inseguro

La mejor forma de proteger un dato confidencial es no guardarlo. Hay datos que en realidad no es necesario almacenar, por ejemplo las contraseñas de usuario. Para esta clase de datos lo que se guarda es una representación no reversible de la contraseña mas no la contraseña como tal. Esta representación se conoce como HASH y es como funciona por defecto el Framework cuando se trata de contraseñas, sin embargo este mismo comportamiento debe aplicarse para datos similares.
Cuando realmente sea necesario guardar el dato la recomendación es utilizar las librerías de cifrado provistas por el Framework o por la plataforma, no desarrollar librerías propias.

URLs privilegiadas sin restricción

La aplicación web tendrá áreas con acceso restringido. Estas áreas se diferencian de otras típicamente por su URL, ej: madr.org.co/Reportes/ProductoresPorGeografia . En el desarrollo se deben proveer los mecanismos adecuados para asegurar que estas áreas solo puedan ser consumidas por usuarios autorizados. El Framework provee

todos los mecanismos necesarios para garantizar esto, sin embargo es necesario que durante la codificación se incluyan en donde correspondan y estén correctamente configurados.

Redirecciones no validadas

A veces las aplicaciones Web redirigen a los usuarios a otros sitios de internet y obtienen recursos estáticos como librerías e imágenes de sitios que no controlan. Debe utilizarse mucha cautela cuando se incluyan vínculos a sitios externos, solo obtener recursos estáticos de sitios conocidos como el CDN y evitar que los mismos usuarios incluyan hipervínculos. Si los usuarios requieren introducir hipervínculos la solución deberá entonces contar con una lista blanca de URLs y dominios permitidos, vetados previamente por un oficial de seguridad o quien haga sus veces.

Identidad, autenticación y autorización

Como se describió en el contexto de negocio se prevén tres grupos de usuarios para las aplicaciones Web del ministerio: los usuarios internos del ministerio, los funcionarios de entidades adscritas que tengan privilegios en el sistema y el público en general.

Usuarios del ministerio

Estos usuarios tienen credenciales del directorio activo del ministerio. Además hacen uso de las aplicaciones Web del ministerio dentro del ministerio lo que les brinda autenticación integrada de Windows. En el caso de utilizar las aplicaciones Web desde internet fuera de la red del ministerio, la aplicación Web les pedirá credenciales pero estas serán las mismas que utilizan en la red.
Respecto a autorización, para aplicaciones Web relacionadas con Dynamics CRM la aplicación Web tomará los roles de Dynamics CRM.
Para aplicaciones que no tengan relación con Dynamics se deben utilizar grupos de seguridad de Active Directory.

Funcionarios de entidades adscritas y otras entidades del sector

Se refiere a usuarios que cuentan con algún privilegio en la aplicación para acceder a cierta área restringida. La identidad de estos usuarios se mantendrá en, o a través de Azure Active Directory y permitirá hacer login social, es decir les permitirá utilizar la misma contraseña con que se autentican en sus redes sociales como Facebook o Twitter.

Para estos usuarios se construirán una serie de páginas de autoservicio, una de ellas para solicitar darse de alta en la aplicación. Esta solicitud corre un flujo de trabajo para

que un funcionario del ministerio apruebe o rechace la solicitud, y al aprobarla le conceda los roles que le correspondan.

Una vez el usuario esté dado de alta podrá ingresar a la aplicación o bien con un usuario y clave propio o a través de su usuario y clave de red social.
Los roles de estos usuarios serán grupos de Azure Active Directory.

Público en general

Las zonas de carácter público de la aplicación no tienen ninguna restricción por lo que no existen usuarios o roles. Existe funcionalidad destinada a los productores donde la aplicación debe poder distinguir un productor de otro, en particular la facilidad de registro y actualización de datos del productor. Para esta funcionalidad no se creará un mecanismo de autenticación basado en usuario y contraseña sino se ofrecerán varias tecnologías para autenticar al usuario, al menos:

- Captura del código de barras de la cédula por cámara Web
- Preguntas de seguridad, ej: Donde fue emitida su cédula?

Desempeño

Dados los escenarios de uso de esta aplicación, donde un número representativo de sus usuarios tiene restricciones de ancho de banda o conectividad, hace que el desempeño percibido por estos pueda ser relativamente bajo así la aplicación y su plataforma provean excelentes tiempos de respuesta.
Por un lado debe procurarse que el tiempo de respuesta al 95% de las peticiones sea menor a 100ms descontando la latencia de red.
Por otro lado debe asegurarse la máxima reducción posible en la cantidad de peticiones necesarias para dar respuesta al usuario y que la solicitud y respuesta a cada petición sea lo más pequeña posible. Para lograr esto se deben aprovechar todas las tecnologías disponibles en los navegadores y en el Framework:

Minification: Hacer los scripts y css lo más pequeños posible eliminado espacios y comentarios y empleado la forma más abreviada disponible del lenguaje

Bundling: Combinar los scripts y los css en un solo archivo reduciendo así la cantidad de requests

Image-Sprite: Combinar los iconos e imágenes pequeñas en un solo archivo y dibujar solo la imagen necesaria mediante css

Inline-Image: Incluir los iconos e imágenes pequeñas como expresiones binarias dentro del propio CSS reduciendo así la cantidad de requests

CDN: Para librerías comunes como jquery, angular, bootstrap y similares utilizar CDNs públicas populares. Al ser la misma URL que potencialmente otro sitio ya ha utilizado el navegador reutiliza la librería descargada previamente.
Esta restricción de ancho de banda y conectividad cierra la puerta a aplicaciones Web tipo RIA y tipo SPA. Las aplicaciones Web para Productores 360 deberán ser HTML5 + Ajax procurando minimizar el tamaño de los recursos que necesitan y la cantidad de llamados para dar respuesta al usuario.

Usabilidad y accesibilidad

Por tratarse de sitios Web del estado Colombiano para sus ciudadanos las aplicaciones Web de productores 360 deben aplicar los lineamientos que el MINTIC ha emitido al respecto a accesibilidad y usabilidad, Ref: http://estrategia.gobiernoenlinea.gov.co/623/w3-propertyvalue-8011.html

Administrabilidad

Todos los aspectos parametrizables de la aplicación Web deben estar expuestos en el Portal de administración de Azure. Esta funcionalidad la brinda automáticamente Azure para el caso de variables de aplicación disponibles en el archivo .config de la aplicación. A nivel de codificación es importante garantizar que no se deje ningún valor “quemado” en código. Para prevenir esto es recomendable no codificar valores por defecto sino permitir que la aplicación falle si hace falta el valor paramétrico.

Soportabilidad

Todos los aspectos de registros, logs de aplicación, alertas de aplicación y de infraestructura deben estar disponibles al menos en el portal de administración de Azure con la posibilidad de brindarlos también a consolas tipo System Center.
Este aspecto se soportará mediante *Application Insights*, ref: http://azure.microsoft.com/en-us/services/application-insights/

Compatibilidad

Tradicionalmente para aplicaciones Web se daba énfasis en brindar compatibilidad a versiones antiguas de los navegadores. Este enfoque hoy en día ha cambiado pues los navegadores antiguos son ahora obsoletos, ya no cuentan con soporte de parte de sus

fabricantes y las principales páginas de internet ya no los soportan. Adicionalmente los navegadores más utilizados se actualizan automáticamente por lo que el concepto de soporte a versiones anteriores deja de ser relevante y ahora se trata de estar preparado para las nuevas versiones.
Las aplicaciones Web del Productores 360 serán compatibles con las últimas versiones y la versión anterior al momento del desarrollo, de los tres navegadores más utilizados en Colombia: Internet Explorer, Mozilla Firefox y Google Chrome, ref:
http://gs.statcounter.com/#desktop-browser-CO-monthly-201405-201505
Un aspecto muy relevante a nivel de tendencia de uso es la popularidad de las tabletas y teléfonos inteligentes como dispositivo para consumo de páginas Web. A nivel mundial en 2015 ya es más común utilizar internet desde esa clase de dispositivos que desde un computador convencional.

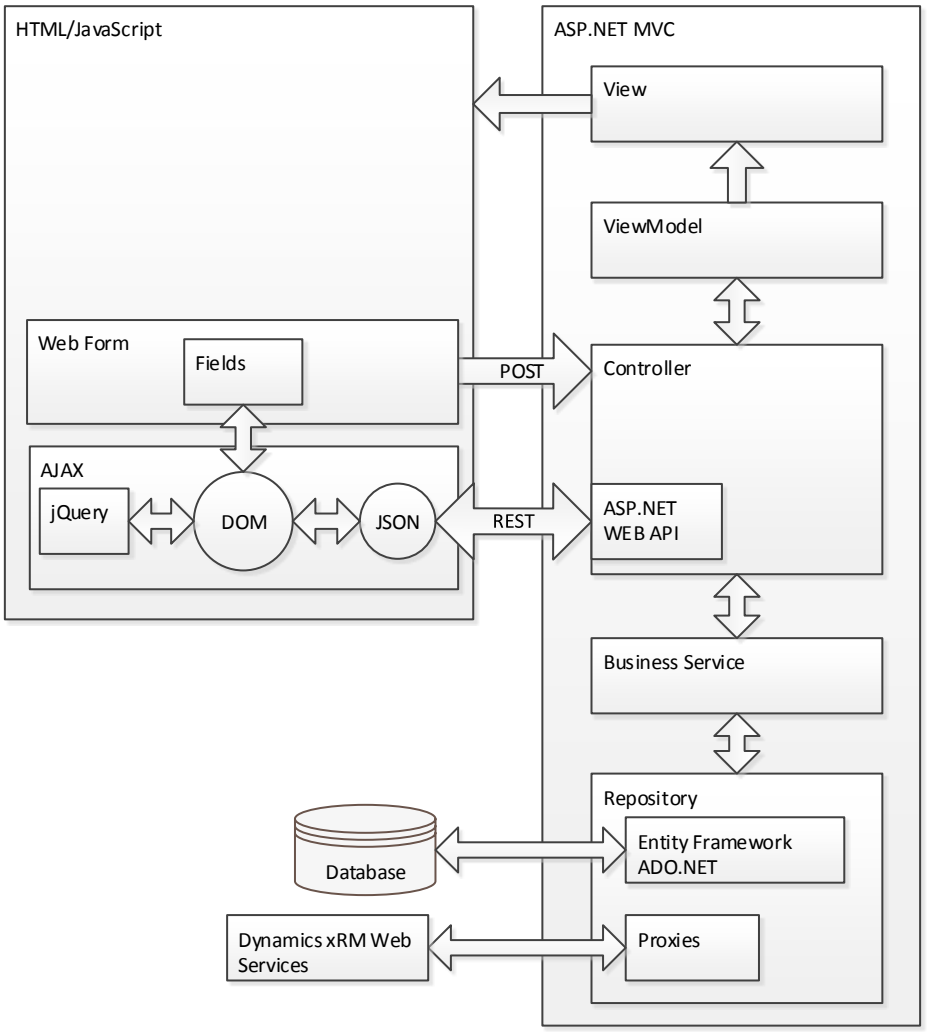
Las aplicaciones Web de productores 360 deben abordar adecuadamente esta tendencia al ajustar la presentación de sus páginas Web cuando se consumen desde un dispositivo móvil. En la implementación debe contemplarse el uso de estilos "*responsive*" que transformen la interfaz de usuario en base a las características del dispositivo desde el que se utiliza. Así mismo los controles para entrada y consumo de datos deben ser amigables con el uso de comandos al toque ("*Touch Friendly*") y aprovechar los controles provistos en las plataformas móviles para brindarles a los usuarios una experiencia óptima.
Ref: https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-deploy-aspnet-mvc-mobile-app/

Los casos de prueba de pruebas manuales y pruebas de interfaz de usuario deben incluir casos de prueba para uso desde dispositivos móviles cuyo criterio de aceptación involucra el correcto ajuste de cada página web al tamaño de la pantalla del dispositivo para los tres navegadores móviles mas utilizados en Colombia, Google Chrome, Android y Safari.
Ref: http://gs.statcounter.com/#mobile+tablet-browser-CO-monthly-201405-201505

6. Arquitectura lógica

ASP.NET MVC

Las aplicaciones Web de Productores 360 se construirán sobre tecnología ASP.NET MVC 5, que es el estándar prevalente para construir aplicaciones modernas robustas y de excelente desempeño. ASP.NET MVC brinda completo control sobre lo que se le entrega al cliente permitiendo aprovechar plenamente todas las tecnologías de internet como HTML5, Javascript y CSS. El patrón de diseño MVC es un patrón establecido y probado por la industria, ideal para tecnologías web.



Una aplicación Web típica de Productores 360 funcionará de la siguiente manera; en el lado cliente se entregará HTML5 y Javascript que ejecutará el navegador. La interfaz de usuario se podrá enriquecer con librerías lado cliente como jquery y knockout, solo cuando HTML5 no sea suficiente.

Pueden existir peticiones asincrónicas al servidor cuando se requerían vía llamados a servicios REST, por ejemplo cuando el usuario esté escogiendo un tipo de producto o una ubicación geográfica. En este punto debe lograrse un balance entre funcionalidad y desempeño dadas las restricciones de ancho de banda.
En el lado servidor, cuando llega una petición desde el navegador, bien sea un FORM POST o un llamado asincrónico REST, el controlador recibe la petición como una

acción. En la gráfica se diferencian los controladores para Web de los de WebAPI (REST) pues el framework brinda tecnologías especializadas separadas para cada uno. La idea es aprovechar esta separación de forma lógica más no física. Los datos enviados se convierten en un ViewModel, que es el objeto con el que trabajará el controlador.

El controlador invoca a los servicios de negocio para que este ejecute el trabajo que se requiera, normalmente leer o escribir datos, o interactuar con otro componente como en este caso Dynamics CRM a través de Web Services. El acceso a estos componentes se encapsula en una capa de repositorio.
Luego de que el controlador ha invocado los componentes necesarios para ejecutar el trabajo actualiza el ViewModel y según el tipo de respuesta de la acción lo envía a la vista adecuada o retorna un objeto JSON. En el caso de vistas estas se componen con el ViewModel y se entrega al navegador para que este la muestre.

Patrón de trabajo centrado en colas

El trabajo resultante de las operaciones de negocio va a consistir en invocar una o varias operaciones en base de datos o en Dynamics CRM. Estas invocaciones pueden tomar mucho tiempo o pueden fallar transitoriamente por intermitencias en la disponibilidad de estos servicios. Si bien para el caso de base de datos los patrones de manejo de fallas transitorias suelen mitigar significativamente este problema este no suele ser el caso en Dynamics CRM.

Para abordar las operaciones diferentes a consultas que dependan de Dynamics CRM se recomienda siempre implementar un patrón de trabajo centrado en colas, donde la operación se limita a agregar el trabajo en una cola y otro proceso se encarga de atender la cola. Desde el punto de vista del usuario su solicitud ya fue recibida pero aún no se sabe el resultado. Con el uso de colas se habilitan escenarios de reintento y se fortalecen varios aspectos de calidad de la aplicación.

Estas colas se implementarán mediante Azure Queues, y el código de referencia de implementación de este patrón está disponible en:
http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/queue-centric-work-pattern

El proceso encargado de atender la cola es un "*WebJob*", que es código .NET que se ejecuta automáticamente por tiempo o basado en eventos, en este caso monitoreando nuevas entradas en la cola. Este "*WebJob*" se despliega junto con el sitio web de la aplicación principal, ref:
https://azure.microsoft.com/en-us/documentation/articles/websites-dotnet-webjobs-sdk-get-started/

Cache

En el caso de operaciones de consulta, cuando se hagan consultas repetitivas debe considerarse el uso de Azure Radis Cache. En el caso de datos relativamente numerosos y que cambian muy poco como los códigos DANE para identificar geografías (DIVIPOLA) y la codificación ICA para productos se recomienda utilizar la estrategia de carga previa del cache (Background data push). Para las demás consultas se recomienda utilizar el cache por demanda.
El código de referencia de implementación para la estrategia por demanda está disponible en:
http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/distributed-caching
En la estrategia de carga previa se utiliza una codificación similar con la diferencia que esta se invoca periódicamente por la misma aplicación, o cuando el proceso de la aplicación inicia, y carga en ese momento todos los valores al cache.

Inyección de dependencias

El uso de un contenedor para inyección de dependencias es mandatorio para facilitar el uso de pruebas unitarias. El contenedor específico a utilizar no se prescribe en esta arquitectura pues no se observa alguna ventaja representativa entre las opciones disponibles, entre ellas *Unity, Autofac y Ninject*. El contenedor específico a utilizar será con el que esté más familiarizado el equipo de desarrollo.

7. Ambiente de desarrollo

Estaciones de trabajo

Todo el desarrollo de estas aplicaciones se realizará utilizando Visual Studio en estaciones de trabajo de desarrollo o en máquinas virtuales de Windows Azure.

Cada estación de trabajo debe brindarle al desarrollador todo lo que necesita para crear y probar la aplicación de forma completamente aislada, sin depender de otros desarrolladores o de servicios compartidos.

Las estaciones de trabajo deben tener:

- Visual Studio 2013 Pro / Premium o Ultimate
- Acceso a internet
- SQL Server 2012 Developer Edition
- Últimas versiones de los navegadores Internet Explorer, Google Chrome y Mozilla Firefox
- Windows Azure SDK
- Dynamics CRM 2015 SDK
- Azure PowerShell Cmdlets
- Web Essentials (Plugin de Visual Studio)

Repositorio, construcción y pruebas

El equipo de desarrollo requerirá además una serie de servicios compartidos:

- Herramienta de ALM – TFS o VisualStudioOnline del ministerio
- Servidor de compilación (Build Server) - TFS o VisualStudioOnline del ministerio
- Ambiente de pruebas – En este ambiente se desplegará de forma continua o al menos diaria el trabajo del equipo de desarrollo para correr pruebas de

integración, pruebas automatizadas y pruebas manuales. Este ambiente estará en Windows Azure.

8. Referencias

En la elaboración de este documento se tuvo en cuenta la literatura enumerada a continuación:

Building Cloud Apps with Azure - http://aka.ms/CloudApps_PDF
.NET Technology Guide for Business Applications - http://aka.ms/NETTechGuide/ebook

Application Architecture Guide, 2nd Edition – https://www.microsoft.com/downloads/details.aspx?FamilyID=ce40e4e1-9838-4c89-a197-a373b2a60df2&DisplayLang=en

9. Control de Versiones

Fecha	Versión	Descripción
25 junio 2015	1	Inicial