EBook Gratis

APRENDIZAJE odata

Free unaffiliated eBook created from
Stack Overflow contributors.

#odata

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: odata

It is an unofficial and free odata ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official odata.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con odata

Observaciones

Esta sección proporciona una descripción general de qué es odata y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de odata, y vincular a los temas relacionados. Dado que la Documentación para odata es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Instalación o configuración

Instrucciones detalladas para configurar o instalar odata.

Odata- La mejor manera de descansar.

De odata.org

> OData (Open Data Protocol) es un estándar de OASIS que define las mejores prácticas para crear y consumir API RESTful. OData lo ayuda a concentrarse en la lógica de su negocio mientras crea API RESTful sin tener que preocuparse por los enfoques para definir encabezados de solicitud y respuesta, códigos de estado, métodos HTTP, convenciones de URL, tipos de medios, formatos de carga útil y opciones de consulta, etc. OData también lo guía. seguimiento de cambios, definición de funciones / acciones para procedimientos reutilizables y envío de solicitudes asíncronas / por lotes, etc. Además, OData proporciona la facilidad de extensión para satisfacer cualquier necesidad personalizada de sus API RESTful.

La característica más emocionante de Odata desde mi punto de vista es,

1. **Convenciones de url**

En las API normales, no hay una forma estándar de especificar una url, lo que significa que al ver a una API no podemos asegurarnos de lo que está haciendo esa API. Odata ayuda a crear una URL estándar basada en la lógica de negocios.

2. **Preguntando**

En la API normal, una vez que creamos más si queremos solo datos específicos de la respuesta, lo haremos así.

- Llame a la API (Desde la API del servidor se devuelve todo)
- Aplicar filtrado en el lado del cliente.

de lo contrario crea una API separada que resulta en datos filtrados.

Pero en lugar de estas opciones, la API de OData permite la opción de consulta significa que podemos incluir las condiciones de filtrado en la url de Odata, Odata filtra automáticamente el resultado del servidor para que podamos obtener los datos que deseamos.

Para jugar con Odata en el cartero http://www.odata.org/getting-started/learning-odata-on-postman/

Lea Empezando con odata en línea: https://riptutorial.com/es/odata/topic/5830/empezando-con-odata

Capítulo 2: Autenticación de Azure AD para Node.js

Introducción

Este documento proporciona una descripción general de alto nivel y explica la arquitectura completa del proceso de autenticación de Azure AD para Node.js (HERRAMIENTA MÓVIL). Explica el componente técnico y su interacción entre la aplicación móvil, la API web, la base de datos de documentos y el Directorio activo de Azure. Finalmente, cómo el usuario móvil podrá iniciar sesión en el sistema y realizar operaciones. El documento proporciona una descripción de alto nivel de los objetivos de la arquitectura, los casos de uso compatibles con el sistema y los estilos arquitectónicos y comp.

Examples

Contenido

Documento de diseño de alto nivel

Herramienta MÓVIL (autenticación de Azure AD para Node.js)

Versión 1.0 (borrador)

Tabla de contenido

Diseño de alto nivel (HLD)

1. Introducción Este documento proporciona una descripción general de alto nivel y explica la arquitectura completa del proceso de autenticación de Azure AD para Node.js (HERRAMIENTA MÓVIL). Explica el componente técnico y su interacción entre la aplicación móvil, la API web, el DB de documentos y el Directorio activo de Azure. Finalmente, cómo el usuario móvil podrá iniciar sesión en el sistema y realizar operaciones. El documento proporciona una descripción de alto nivel de los objetivos de la arquitectura, los casos de uso compatibles con el sistema y los estilos y componentes arquitectónicos que se han seleccionado para lograr los mejores casos de uso. 1.1. El propósito del documento de diseño de alto nivel (HLD) proporciona una visión general de la arquitectura de la aplicación móvil. 1.2. Alcance El alcance de este HLD es representar la arquitectura de la APLICACIÓN MÓVIL desarrollada por el equipo de Preludesys.

1.3. Definiciones, acrónimos y abreviaturas

- HTTP - Protocolo de transferencia de hipertexto
- Azure AD - Azure Active Directory
- Web-API - Interfaz de programa de aplicación web
- API REST - Transferencia de estado representacional
- JSON - Notación de objetos de JavaScript
- ADAL - Biblioteca de autenticación de Active Directory de Azure
- iOS - Sistema operativo para dispositivos Apple
- Android - Sistema operativo para dispositivos de Google
- OAuth2 / OpenID- un estándar abierto para la autorización
- Usuario: este es un usuario de la organización que tiene un miembro de Azure Active Directory.
- Creador (propietario del proceso): este es un usuario que puede crear / modificar un documento
- Reader - Este usuario puede leer / ver datos
- Administrador: este usuario puede leer, modificar y eliminar cualquier Especificación de proceso, Revisión de proceso, Plantilla de proceso y herramienta de DTCPII y administrar otros derechos / roles de usuario. El administrador puede delegar o compartir derechos administrativos a otros usuarios en el sistema.

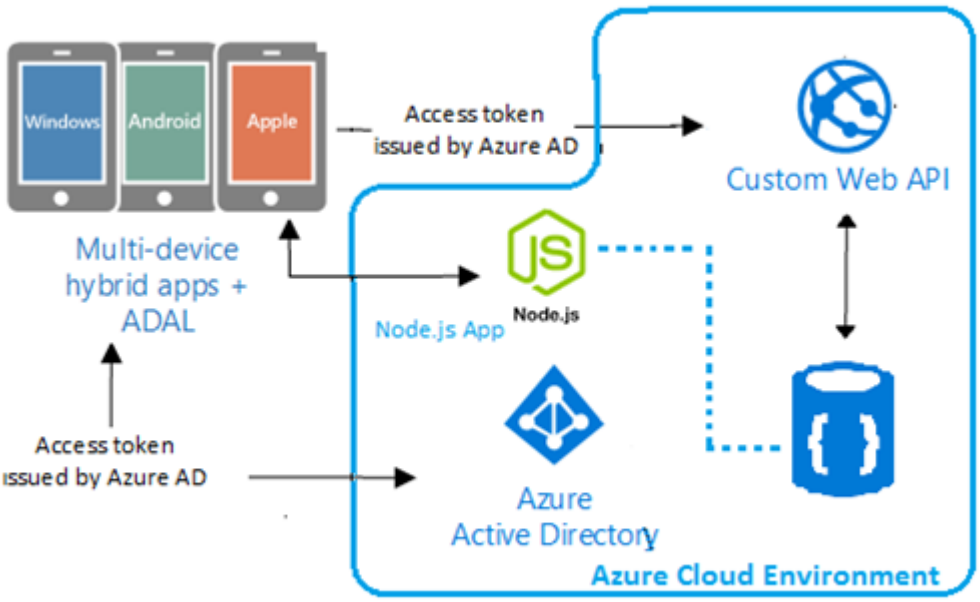
1.4. Referencias

1.5. Visión general

Este documento proporciona una descripción técnica de los siguientes componentes.

- Módulo Node.JS
- API web personalizada
- Biblioteca ADAL
- Active Directory de Azure
- Documento DB

2. Representación arquitectónica (vista lógica)

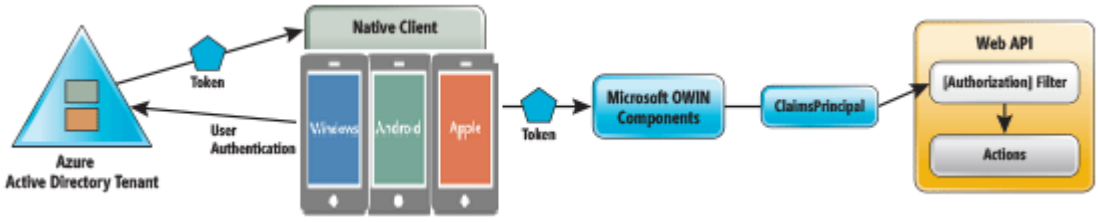


La solución se divide en dos componentes. Uno es la aplicación del lado del cliente que se ejecutaría en el componente móvil y del lado del servidor que se ejecuta en el entorno de Azure.

Aplicación móvil (Cliente Node.js) Node.js es una interfaz de usuario ligera y sensible. Una aplicación móvil sería simplemente, escrita en el idioma nativo o utilizando una solución híbrida como Cordova (también conocida como PhoneGap). Esta interfaz de usuario obtendría datos y realizaría la lógica comercial mediante llamadas a una API del lado del servidor. Si la aplicación móvil requiere partes de la lógica empresarial, tendría que ser reescrita en object-C para iOS y Java para Android. Si bien Cordova ofrece una solución para la reutilización de algunos códigos, no puede aprovechar Node o su ecosistema directamente.

Server Side Component Node.js Server Node es un tiempo de ejecución de servidor web JavaScript. Esto actúa como servidor web para aplicaciones móviles. Utiliza JSON ya que el formato de intercambio de datos, JSON, se puede analizar de forma nativa en ambos extremos. El nodo difiere de la mayoría de los tiempos de ejecución de las aplicaciones web en la forma en que maneja la concurrencia. En lugar de utilizar subprocesos para lograr la concurrencia, Node se basa en un bucle controlado por eventos que se ejecuta en un solo proceso. El nodo admite un modelo asíncrono (sin bloqueo). API web Esta es una solución de API REST que se ejecuta en Azure como API web. Se ejecuta sobre Document DB y actúa como capa protectora para Document DB. Las herramientas ASP.NET y el middleware de seguridad de los componentes de la Interfaz Web Abierta de Microsoft para .NET (OWIN) hacen que la protección de la API Web sea sencilla. El punto final está protegido mediante Azure AD OAuth Bearer 2.0 Access Tokens.

3. Proceso tecnico



3.1. Autenticación La autenticación se reduce a pedirle a la persona que llama, al enviar un

mensaje a un servidor, que incluya algún tipo de credencial que pueda verificar su identidad o recuperar sus atributos. Luego, el servidor utiliza esa información para la autorización, lo que determina si se debe conceder acceso y en qué términos. Los recursos a menudo descargan la mayoría de las funciones de autenticación a un proveedor de servicios externo, comúnmente conocido como una autoridad o un proveedor de identidad. Estos proveedores se encargan de tareas pesadas como la incorporación de usuarios, la asignación de credenciales, el manejo de flujos de ciclo de vida (como la recuperación de contraseñas), el suministro de una IU para la autenticación de usuarios, la verificación de credenciales a través de múltiples protocolos, la administración de múltiples factores de autenticación, la detección de fraudes y más. Con esas funciones fuera del camino, la única tarea de autenticación que queda es verificar que la autenticación haya tenido éxito en la autoridad de elección. Normalmente, esto implica examinar un token de seguridad, un fragmento de datos emitido por una autoridad a una persona que llama luego de una autenticación exitosa. Los tokens de seguridad generalmente se diseñan de acuerdo con un formato específico, firmado digitalmente por una clave que identificará inequívocamente a la autoridad emisora, y contendrá algunos datos que vinculan de forma única el token al recurso de destino. Cuando el recurso recibe una solicitud, busca un token acompañante. Si encuentra uno que cumpla con los atributos de validación requeridos, la persona que llama se autentica. En ese nivel de detalle, el patrón es tan genérico que puede describir muchos enfoques diferentes para la autenticación. Lo aplicaré a este escenario asignando los roles de alto nivel a entidades concretas. El recurso El recurso será el proyecto ASP.NET Web API 2 que necesito asegurar. Puede aplicar los requisitos de autenticación con una granularidad más fina.

3.2. API web personalizada y punto final seguro La Autoridad configurará la API web para descargar sus necesidades de autenticación a Windows Azure AD, una plataforma como servicio (PaaS) disponible para cada suscriptor de Windows Azure. Windows Azure AD está diseñado específicamente para soportar cargas de trabajo de aplicaciones basadas en la nube. El servicio almacena información sobre los usuarios (atributos y credenciales) y la estructura organizativa. Puede sincronizar datos con Windows Server Active Directory, siempre que se conecte exclusivamente en la nube. 3.3. Formato de token de OAuth El proceso de validación de OAuth 2.0 no exige ningún formato de token en particular, pero el formato de token web JSON (JWT) (bit.ly/14EhlE8) para escenarios REST se ha convertido en el estándar de facto. Tanto Windows Azure AD como los componentes de Microsoft OWIN son compatibles con el formato JWT en los flujos de OAuth 2.0. Los mecanismos de adquisición y validación de JWT están a cargo del middleware, y el formato del token es transparente al código de la aplicación. 3.4. El módulo Node.js con la biblioteca ADAL Los módulos Node.js es una interfaz móvil que contiene componentes de UI y un módulo de autenticación para obtener el token de seguridad de Active Directory mediante la biblioteca ADAL. La biblioteca ADAL para node.js facilita que las aplicaciones node.js se autentiquen en Azure AD para acceder a los recursos web protegidos de Azure AD.

3.5. Configuración y configuración de base de datos de documentos

3.6. Sincronización de Azure AD con Azure AD Connect Esto está fuera del alcance, ya que los usuarios empresariales ya forman parte de office 365, lo que implica que ya están sincronizados con Azure Active Directory.

4. Metas y restricciones arquitectónicas del lado del servidor La herramienta móvil se alojará en la plataforma Azure Cloud que se ejecuta como aplicación web. Toda la comunicación con el cliente debe cumplir con los estándares de protocolo de comunicación HTTPS, TCP / IP. Los usuarios del lado del cliente podrán acceder a la herramienta móvil a través de dispositivos Android e iOS.

4.1. Seguridad

Para poder autenticar a los usuarios, debe registrar la aplicación Node.js con Azure Active Directory (AAD). Esto se hace en dos pasos. Primero, debe registrar su servicio móvil y exponer los permisos en él. En segundo lugar, debe registrar su aplicación iOS y concederle acceso a esos permisos 4.2. Persistencia

La persistencia de los datos se tratará utilizando la base de datos de documentos de Azure.

4.3. Fiabilidad / Disponibilidad

4.4. Actuación

No hay restricciones particulares relacionadas con el rendimiento del sistema. Se anticipa que el sistema debe responder a cualquier solicitud bajo los tiempos de espera estándar de la base de datos y del servidor web (20 segundos), además el rendimiento del sistema puede depender del hardware disponible, la red de PSU y las capacidades de conexión a Internet. Además, los tiempos de carga / descarga pueden depender del tamaño de los datos, que a su vez depende de la entrada del usuario. Por lo tanto, el rendimiento real se puede determinar solo después de la implementación y prueba del sistema.

5. Tamaño y rendimiento

6. Problemas y preocupaciones

Lea Autenticación de Azure AD para Node.js en línea:
https://riptutorial.com/es/odata/topic/10583/autenticacion-de-azure-ad-para-node-js

Creditos

S. No	Capítulos	Contributors
1	Empezando con odata	Community, LMK
2	Autenticación de Azure AD para Node.js	Sandip Dalvi