ARQUITECTURA DE REFERENCIA PARA APLICACIONES WEB

Última actualización: Febrero de 2022

Gestión Informática
Desarrollo Institucional
Universidad de Antioquia

CONTROL DE CAMBIOS

Fecha	Autor	Descripción
Agosto 20 de 2020	Andrés López	Ajuste a la página de monitoreo de la aplicación, monitoreo de integraciones y ajuste de seguridad en capa de interfaz de usuario.
Mayo 11 de 2021	Andrés López	Se agregan las secciones: Seguridad, Tablas altamente transaccionales y Código de respuesta para las servicios web.
Junio 4 de 2021	Alexandra Marin	Se especifica la librería logback de uso estándar para gestionar los logs en las aplicaciones
Diciembre de 2021	Alexandra Marín	Se actualizan algunos vínculos.
Febrero de 2022	Alexandra Marín	Se actualiza la referencia al archivo de Estándares de línea gráfica y versión de angular.

CONTENIDO

CAPAS DE LA ARQUITECTURA DE REFERENCIA

El documento de referencia está dividido en las siguientes capas:

Capa de Interfaz de usuario: estándares y herramientas para soportar la interfaz de usuario y/o los servicios de presentación. (p.e: html, interfaces de usuario enriquecidas).

Capa de negocio: estándares y herramientas para soportar la lógica de negocio. (p.e: lenguajes, estándares de componentes).

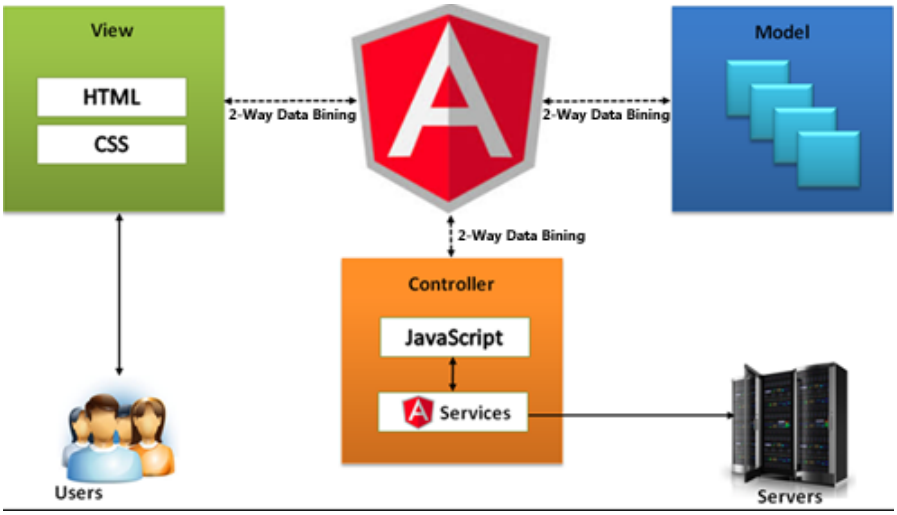
Capa media: estándares y herramientas para soportar la gestión de transacciones y la comunicación entre procesos. (p.e: servidores de aplicaciones, APIs de IPC).

Capa de software de sistema: estándares y herramientas para soportar la gestión de sistemas. (p.e: sistemas operativos y sistemas de gestión de bases de datos).

A continuación, se detalla cada una de las capas:

Capa de interfaz de usuario

Área	Productos/servicios/componentes
Estilos y usabilidad	Ver el documento “Estandarización de línea grafica.doc” que se encuentra en Google Drive Unidades compartidas > Arquitectura de sofware: https://docs.google.com/document/d/1NCxK4ulBa2bpf2N3LiKRImi9hXjO9BsCDcLVbpwpYZ4/edit?usp=sharing . Este documento define los estándares de línea gráfica que deben seguirse para el desarrollo de aplicaciones web.
Herramientas de construcción	
Lenguajes	Para aplicaciones nuevas web se debe usar Angular en su versión actual más estable. Tener en cuenta que, para iniciar un desarrollo nuevo, es necesario solicitar la plantilla udea-plantilla-frontend a Gestión Informática. Aquí es necesario tener en cuenta que según el tamaño de la aplicación a desarrollar, lo más recomendable es hacer varios módulos front end, por ejemplo uno para reportes, otro para lo administrativo, otro para lo funcional.Sin embargo esta es una decisión a analizar en conjunto con los arquitectos de acuerdo con el análisis de requisitos que se tenga. Para aplicaciones preexistentes en lenguaje PHP 4 y 5, el mantenimiento se debe realizar con el mismo lenguaje y versión. (Solo mantenimiento).

Entorno de desarrollo (IDE)	1. Para aplicaciones que utilicen AngularJS, se recomienda establecer un proyecto en Eclipse y que esté a su vez esté vinculado al repositorio de código fuente institucional. (GitLab 8.0), el estándar de versionamiento se explica en el anexo 1: Estándar_DefinicionDeVersionamiento_UdeA, el plugin: maven-release-plugin que permite generar el archivo desplegable y dejarlo en el repositorio de administración de Software Nexus. 2. Para aplicaciones que utilicen Angular 6 o posteriores, Visual Studio Code. En este IDE instalar las extensiones: Angular Snippets para reconocer todas las etiquetas de angular,Angular Essentials,Angular Files 3. Para php: Dado que en PHP se hacen solo modificaciones a aplicaciones existentes, debe utilizarse lo que actualmente se tiene.
Estrategias de diseño	El framework angular implementa el patrón de diseño MVC 
Seguridad	Todas las páginas deben utilizar el protocolo https y por ende todos los componentes usados en estas, con el fin de no generar en el navegador del usuario alertas de seguridad y errores de ejecución. La Universidad dispone de un servidor de aplicaciones con certificado de seguridad, donde se instalarán las aplicaciones que requieran de dicho protocolo, y de un servidor de pruebas con un certificado no válido. **IMPORTANTE**: La aplicación siempre debe cargar antes de una petición un spinner, y mostrar el contenido sólo cuando el filtro haya permitido la transacción y los datos estén listos para mostrar. Si el filtro no permite la transacción, se debe mostrar que no tiene permiso.

Capa de negocio

Área	Productos/servicios/componentes
Componentes	
1. Lenguajes	1. Java (1.8) o posterior según se indique.

	Tener en cuenta que para el desarrollo se tiene definida una estructura a través de una plantilla y para iniciar, debe solicitar dicha plantilla al equipo de Gestión Informática (plantilla-udea-backend). 2. PHP 5 para mantenimiento de aplicativos hechos en este lenguaje.
2. Entorno de desarrollo (IDE)	Se debe usar el IDE Eclipse Neon o posteriores 4. Tener en cuenta que el código fuente debe estar montado en el repositorio de versiones GitLab 8.0 (ver el estándar de versionamiento, anexo 1: Estándar_DefinicionDeVersionamiento_UdeA), y utilizar el plugin: maven-release-plugin que permite generar el archivo desplegable y dejarlo en el repositorio de administración de Software Nexus.
3. Cantidad de módulos	Para la implementación del proyecto se pueden definir uno o varios módulos que finalmente serán uno o varios WAR. Esto dependerá del tamaño de la aplicación y de los requisitos que ésta tenga. Inicialmente se puede pensar en un módulo para lo funcional y en otro para consultas, sin embargo, puede ser uno solo, vale la pena analizarlo y tomar la decisión al momento de iniciar.
Uso de patrones	
Patrón DAO	Patrón recomendado por J2EE https://www.oracle.com/java/technologies/dataaccessobject.html para acceso a datos. Este patrón aplica tanto para aplicaciones Java como para aplicaciones PHP.
Patrón Factory	Patrón recomendado por J2EE para complementar el patrón DAO. Este patrón se usa para dar la posibilidad de acceder a los datos a través de JDBC. La configuración del acceso debe hacerse a través de DataSource, por ende, la aplicación no debe almacenar contraseñas de acceso a base de datos. La url de referencia es la misma que para el patrón DAO. Este patrón aplica tanto para aplicaciones Java como para aplicaciones PHP.
Patrón DTO	Patrón recomendado por J2EE https://www.oracle.com/java/technologies/transfer-object.html para transferencia de datos. Este patrón se usa transversal a toda la aplicación. Este patrón aplica tanto para aplicaciones Java como para aplicaciones PHP.
Componentes de servicio	
1. Seguridad	El MUA (Módulo Único de Autorización) a través del filtro de seguridad, es el encargado de la autorización para acceder a los recursos protegidos (no públicos). Este módulo permite definir la aplicación, roles, opciones de menú y qué roles pueden acceder a dichas opciones. Y asignar roles a los usuarios.
1. Manejo de logs	Se debe utilizar la librería logback [1] en su versión más reciente e incluirla en el propio war de la aplicación. Dado que es como la evolución de log4j, su configuración es muy similar. Debe declararse en el pom y la configuración debe hacerse en el archivo logback.xml que debe estar dentro de la carpeta resources. Los niveles de log son

[1] Logback es una componente de registro de código, abierto, diseñado por el fundador de log4j y que reemplaza

	TRACE, DEBUG, INFO, WARN, ERROR. En los ambientes de desarrollo y pruebas no hay restricción en el nivel, pero para producción, debe quedar en ERROR.
2. Lógica de negocio	La lógica de negocio se debe manejar a través de clases dentro del paquete "**bl" (business logic).**
3. Servicios web	Las clases clientes de servicios web, deben ubicarse dentro del paquete "**bl"**. Para el consumo de servicios web proveídos por la institución se debe usar la librería wsClient.jar tal como se indica en el manual "Para consumir servicios web UdeA. docx". En caso de que la aplicación no sea java, se deben utilizar las direcciones de los ws de forma dinámica a través de la dirección https://link.udea.edu.co/listadows&serviceid=xxx, donde xxx corresponde al nombre del servicio web que se desea consumir y el cual puede ser encontrado en la página https://link.udea.edu.co/listadows.
4. Enrutamiento de peticiones desde la interfaz de usuario hacia la lógica de negocio	Se debe realizar a través de servicios REST.
5. Filtros	Cualquier filtro necesario en la aplicación, deberá ubicarse en el paquete "**flt**".
Componentes de acceso a datos	
La aplicación deberá usar un usuario específico que no sea el dueño del esquema propio correspondiente. El estándar es nombreAplicacion**App** para las transacciones. A dichos usuarios se les deben otorgar los permisos necesarios para la ejecución de las consultas, operaciones, procedimientos y funciones creados, que utilice la aplicación.	
1. Accesos directos	● Se podrán realizar operaciones directas (CRUD) sobre las tablas que se consideran "Maestras", es decir tablas de configuraciones y listados de valores. ● Se podrán realizar accesos de consulta "select" sobre tablas siempre y cuando no existan datos sensibles y que no sean usados en la aplicación. Ejemplo: Si la aplicación requiere consultar la información personal de un empleado, pero en la tabla también existe el salario y otra información personal sensible, el usuario de base de datos no podrá hacer la consulta directamente, sino a través de una función o una vista. ● Cuando sea necesario cruzar varias tablas, es aconsejable crear vistas, procedimientos y/o funciones. En el caso de usar funciones o procedimientos que retornen más de un registro, se recomienda el uso de **refcursor**, para que no sea necesario el uso de objetos tipados.
2. Procedimientos almacenados	Con el fin de hacer más segura la aplicación y sus transacciones, las operaciones (actualización, creación y eliminación) sobre la base de datos para tablas propias de negocio, deben realizarse a través de procedimientos almacenados y funciones, a no ser que sean las operaciones CRUD sobre las tablas maestras.
3. Tipos de acceso a datos	Utilizar JDBC (Gestionado a través del servidor de aplicaciones con DataSources). Sin embargo, durante la etapa de diseño de la aplicación se

	pueden definir otros ORMs o librerías para el acceso a datos según las necesidades. La configuración del acceso debe hacerse a través de DataSource, por ende la aplicación no debe almacenar contraseñas de acceso a base de datos y debe quedar registrado en el archivo web.xml, desde el cual debe ser utilizado por la aplicación.

Capa media

Área	Productos/servicios/componentes
Servidores de aplicaciones	Para aplicaciones web Java, WAS ND 8.5.5 Tener en cuenta que el ambiente productivo es de alta disponibilidad: Balanceador y dos nodos (clúster). En ambiente de pruebas se tiene disponible una instalación estándar, no en clúster.
Servicios de directorio	Oracle Internet Directory (OID) – En proceso de migración hacia Active Directory.

Capa de software de sistema

Área	Productos/servicios/componentes
Sistemas operativos	Estaciones para desarrollo: Microsoft Windows 7 y posteriores.
SGBD	Oracle 11g (En proceso de actualizacion)

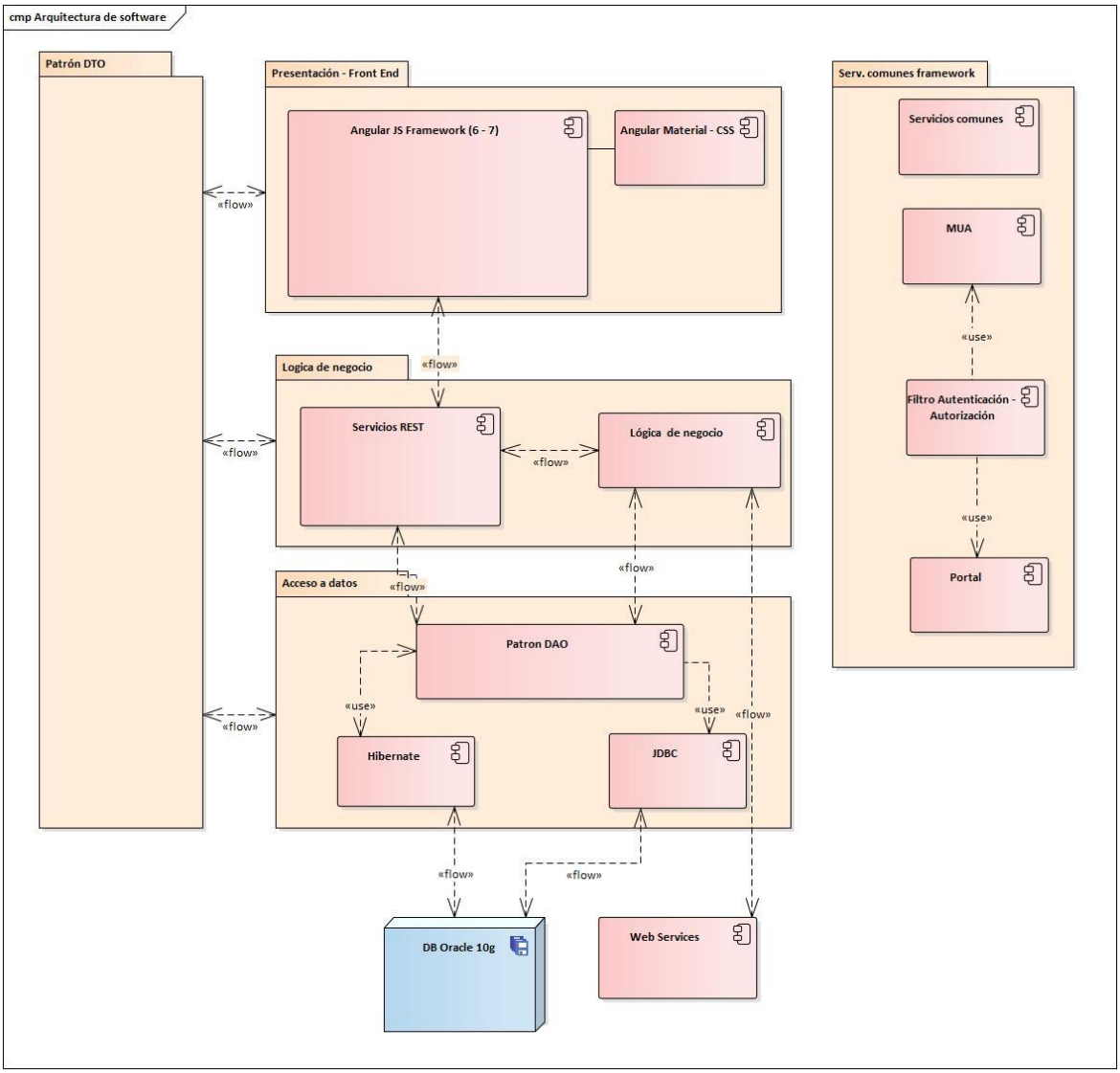
AMBIENTES

Desarrollo: PC de cada desarrollador con Tomcat y base de datos en ambiente de desarrollo Oracle, también se tiene la posibilidad de un ambiente WAS para desarrollo (No clusterizado) donde el desarrollador podrá probar sus despliegues antes de enviarlos a ambiente de pruebas, este se llama SVRDLLO.

Pruebas: Servidor aplicaciones de pruebas WAS ND 8.5.5. (Clusterizado) CLUDEATST, acceso a base de datos en ambiente de pruebas, Se tiene disponibilidad de servicios web en ambiente de pruebas, aplicación MUA en ambiente de pruebas.

Producción: Servidor de aplicaciones WAS ND 8.5.5, Alta disponibilidad (2 nodos) (cluster) CLUDEAPDN, aplicación MUA, Servicios WEB, base de datos de producción.

DIAGRAMA GENERAL PARA DESARROLLO DE APLICACIONES WEB EN JAVA CON ANGULAR



OTROS COMPONENTES

Analíticas de la aplicación

Las aplicaciones que generan presentación HTML deberán tener manejo de analíticas a través de Google, para lo cual se deberá consultar la siguiente tabla el código para la aplicación, en caso de no existir, solicitar la entrega del código a través del correo andres.lopez@udea.edu.co y dennis.marin@udea.edu.co, indicando la dirección web de la aplicación. Una vez tenga el código, deberá agregar incorporarlo en el siguiente código, y este a su vez en la aplicación para que toda página HTML lo tenga inmerso:

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-135941959-31"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-135941959-31');
</script>
```

Códigos existentes
El listado vigente se puede consultar en la siguiente dirección:
https://drive.google.com/open?id=1Qi7d46WW5eYZzOG0OMGhdlc7nmTCbemm-6CcQHjgHFc

IMPORTANTE: Tener presente que estos códigos sólo funcionarán en la dirección indicada.

Almacenamiento de archivos y logs

Las aplicaciones ya sea en producción o pruebas, deberán almacenar los archivos que los usuarios suban o que esta genere dinámicamente en una carpeta diferente a la de despliegue de la aplicación, esto con el fin de evitar pérdida de archivos por motivo de redespliegues. La carpeta será: ***/web/nfs/ArchivosOYS/<nombreApp>***.

Para el caso de los logs, estos se deberán almacenarán en la carpeta: ***/web/nfs/ArchivosOYS/logs***.

Las carpetas disponibles para archivos de aplicaciones en producción y pruebas pueden ser consultado en la siguiente dirección:

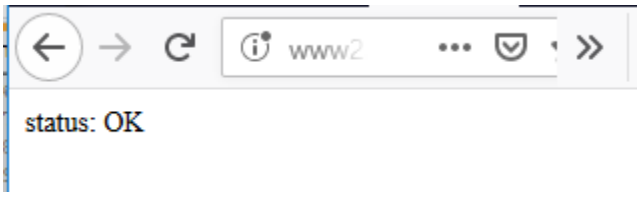
https://docs.google.com/spreadsheets/d/1RKTKKPnKJ2Ko1CpfzPA3MGv9H8a-f5mpM0UnZZJcsvk/edit?usp=sharing

Si su aplicación no tiene una carpeta asignada deberá ponerse en contacto con alguno de los arquitectos.

En algunos casos, donde la aplicación en modo prueba estará desplegada en producción, se deberá usar otra carpeta para evitar que utilice los archivos productivos, para lo cual deberá ponerse en contacto con uno de los arquitectos.

Página de monitoreo de la aplicación

Con el fin de poder monitorear la aplicación tanto en su funcionamiento con en conexión con la base de datos a través de un agente (actualmente Zabbix), se deberá tener en cada aplicación un recurso, ya sea página, servlet, webservice, o el que aplique para la tecnología usada en la aplicación, que haga conexión a la base de datos por defecto de la aplicación, ejecute la consulta: ***select monitoreodb.mon_status_conexion status from dual*** y muestre el resultado en la página. Esta funcionalidad es de carácter público y retorna un texto predeterminado, a través del cual se podrá conocer que efectivamente la aplicación se está conectando a la base de datos. Finalmente, la página deberá estar exenta de cualquier seguridad a nivel de usuario autenticado, excluida en el filtro de seguridad y de ser posible, ser plana, es decir, sin estilos ni formato, sólo mostrará la respuesta retornada por la base de datos, tal como aparece en la siguiente imagen.



Para el caso de aplicaciones exclusivamente servicios web, esta página debe corresponder a un WS, con las mismas características definidas en el párrafo anterior.

Página de monitoreo de integraciones

Si la aplicación tiene integraciones con aplicaciones externas a través de peticiones http/https, deberá existir una funcionalidad adicional que permita monitorear cada una de las integraciones. La idea de esta funcionalidad, es tener un agente que monitoree esta página de forma permanente, así conocer cuál es su estado y tener un histórico. Este monitoreo debe retornar para cada integración un OK en caso exitoso, de lo contrario el error que está ocurriendo. Finalmente, la página deberá estar exenta de cualquier seguridad a nivel de usuario autenticado, excluida en el filtro de seguridad y de ser posible, ser plana, es decir, sin estilos ni formato.

Caso de ejemplo: Si la aplicación tiene una integración con una pasarela de pagos y con un servicio de firma electrónica, entonces deberá existir una página o WS que permita conocer el estado de la integración de cada una de ellas:

Pasarela: https://asone.udea.edu.co/aplicacion/integraciones?id=pasarela

Firma electrónica: https://asone.udea.edu.co/aplicacion/integraciones?id=firma

Para el caso de aplicaciones exclusivamente servicios web, esta página debe corresponder a un WS, con las mismas características definidas en el párrafo anterior.

Monitoreo de logs

Las aplicaciones en lenguaje JAVA, versión 1.6 o superior, en cualquier periodo de estabilización, ya sea la primera salida o un cambio significativo, deberán incorporar el plugin JavaMelody, que permite ver los logs de error de la aplicación, su nivel de ocurrencia, entre otras facilidades, por consiguiente, hacer una oportuna y eficaz gestión de los incidentes. La incorporación de acuerdo a su sitio web (*https://github.com/javamelody/javamelody/wiki*), se hace de la siguiente forma:

1. **Jar files:** Copy the files javamelody.jar and jrobin-1.5.9.jar to the WEB-INF/lib directory of the war of the webapp to monitor. Or if you use Maven, add the javamelody-core dependency in the pom.xml file of your webapp.

2. **web.xml file**: If your application server is compatible with Servlet API 3.0 (like tomcat 7, glassfish v3 or jboss 6), this paragraph is generally not needed, you can skip it and launch the server as in the next paragraph, except if you use a web.xml file without version="3.0". Otherwise add the following lines in the file WEB-INF/web.xml of the war of the webapp, before the description of your servlet:

```
<filter>
        <filter-name>javamelody</filter-name>
        <filter-class>net.bull.javamelody.MonitoringFilter</filter-class>
        <async-supported>true</async-supported>
</filter>
<filter-mapping>
        <filter-name>javamelody</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>ASYNC</dispatcher>
</filter-mapping>
<listener>
        <listener-class>net.bull.javamelody.SessionListener</listener-class>
</listener>
```

<async-supported>true</async-supported> and <dispatcher>ASYNC</dispatcher> are needed to support asynchronous requests with Servlet API 3.0.

3. **First results:** You can now view the monitoring: deploy the war and open the following page in a web navigator after starting the server:

http://<host>/<context>/monitoring

Aplicaciones seguras

Las aplicaciones deben ser desarrolladas de tal forma, que puedan ser habilitadas bajo el protocolo HTTPS, así inicialmente sean desplegadas en HTTPS. Para lograr esto, los recursos que utiliza la

página, cómo js, imágenes, estilos e integraciones con otras aplicaciones, deben ser con rutas relativas, seguras o sin protocolo. En otras palabras así:

1. Los recursos de la propia aplicación deben ser enlazados de forma relativa. De esta forma siempre son cargados a través del mismo protocolo de la página. Ejemplo:

   carpeta/archivo.extensión

2. Los recursos externos deben ser enlazados por https. De esta forma siempre son cargados por https. Ejemplo:

   https://www.sitio.com/*carpeta/archivo.extensión*

3. Los recursos externos deben ser enlazados sin protocolo. De esta forma siempre son cargados a través del mismo protocolo de la página. Ejemplo:

   //www.sitio.com/carpeta/archivo.extensión

Lo anterior aplica siempre y cuando los recursos externos estén disponibles bajo https.

Para las aplicaciones Angular, el frontEnd debe redireccionar siempre hacía https en caso de que la petición sea http, para el backend, sólo debe responder peticiones https y no debe redireccionar las peticiones http, estas debe rechazarlas.

Seguridad

Toda aquella aplicación que tenga opciones que requieran un nivel de seguridad implicando autenticación o manejo de roles, deberá contar con un listado donde se especifique cada opción con su respectiva URL y el ROL del MUA (Módulo Único de Autorización) o GRUPO del OID (LDAP) que deberá tener el usuario para poder acceder. Ejemplo: Si la aplicación tiene opciones que estarán expuestas en el portal para estudiantes, otras para docentes y otras para vicedecanos, estas opciones deberán estar en el listado, para las del portal se debe indicar que serán los GRUPOS UdeAEstudiantes y UdeADocentes respectivamente, mientras que para la otra, se indicará que es para el ROL maresVicedecano del MUA.

Lo anterior aplica para todo tipo de aplicación donde el usuario accede a ella, sea FrontEnd o BackEnd, con lo que se estaría logrando un nivel de seguridad mínimo para todas las aplicaciones, y que si un usuario accede directamente a la aplicación, el filtro de seguridad controlará su acceso.

Tablas altamente transaccionales

Este capítulo se refiere a una propuesta de manejo de aplicaciones que tienen una alta transaccionalidad en algunas de sus tablas, por lo que se propone una forma de manejo. Para ver el detalle, puedes acceder a la dirección:

https://docs.google.com/document/d/1J950Neg8zUDcqJLs3qrOARTyuvYKnrI4epUivmeHa08

Código de respuesta para las servicios web

Toda aplicación que tenga servicios web, ya sea porque los usa en su propio FrontEnd o porque son consumidos por otras aplicaciones, deberán seguir el siguiente listado de códigos HTTP para sus respuestas:

Para transacciones (insert, delete, update):

- 400: Cuando hay un error en la validación de datos de entrada
- 500: Cuando hay un error en la ejecución, ya sea a nivel del programa o en base de datos
- 200 -> ejecución sin problemas

para consultas

- 400: Cuando hay un error en la validación de datos de entrada
- 500: Cuando hay un error en la ejecución, ya sea a nivel del programa o en base de datos
- 200 -> ejecución sin problemas o consulta sin resultados

Es importante que toda respuesta vaya acompañada de la cabecera MIME adecuada tanto para los servicios web como para las demás respuestas que existan en las aplicaciones. Ejemplo: si la respuesta es una página html, entonces text/html, si es un JSON, entonces application/json. El listado completo de MIME Types se puede consultar en:

https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types