Taller 01
Representación de una arquitectura desde diferentes perspectivas utilizando el modelo C4

El taller tiene como objetivo que el estudiante entienda los siguientes aspectos: (i) la arquitectura debe ser representada desde varias perspectivas, (ii) las entradas, actividades y salidas de la etapa de diseño, y (iii) cómo representar la arquitectura mediante el modelo C4.

1.1 Descripción del contexto

Ha sido contratado por el departamento de sistemas de la FIET para desarrollar una aplicación web de página única, y una app que permita la gestión de las actividades administrativas de la maestría en Computación. El coordinador de la maestría podrá gestionar estudiantes, docentes, asignaturas, movilidades, prácticas, generar un reporte de los estudiantes matriculados en los cursos ofertados para un determinado periodo, generar hoja de vida y enviar correos electrónicos. Por otra parte, el estudiante de la maestría podrá realizar las evaluaciones docentes.

1.2 Requisitos funcionales:

A. Yo como coordinador de la maestría quiero iniciar sesión en el sistema para acceder a las diferentes funcionalidades propias de mi rol.
B. Yo como coordinador de la maestría quiero recuperar la contraseña para poder iniciar sesión en el sistema.
C. Yo como coordinador de la maestría quiero modificar mis datos personales para que los reportes queden actualizados con los datos del coordinador actual.
D. Yo como coordinador de la maestría quiero registrar un nuevo estudiante para poder matricularlo en un curso, gestionar sus solicitudes y asociarlo a un proyecto de investigación.
E. Yo como coordinador de la maestría quiero ver los estados de un estudiante durante el desarrollo de la maestría para visualizar el semestre académico y financiero en el cual se encuentra.
F. Yo como coordinador de la maestría quiero registrar una prórroga/suspensión de términos aprobada para un estudiante para almacenar las prórrogas/suspensiones de términos que un estudiante ha tenido durante el desarrollo de la maestría en computación
G. Yo como coordinador de la maestría quiero registrar un nuevo docente para poder asociarlo a la dirección de un estudiante y un curso de la maestría en computación
H. Yo como coordinador de la maestría quiero registrar una asignatura para asociarla a un curso y matricular los estudiantes

I. Yo como coordinador de la maestría quiero generar un reporte de los estudiantes matriculados en los cursos ofertados para un determinado periodo para enviarlo a los directores, docentes y estudiantes
J. Yo como coordinador quiero que la aplicación envíe automáticamente un correo a los estudiantes matriculados en los cursos y directores de los estudiantes informando a cada estudiante y director de cada estudiante, sobre los cursos matriculados por el estudiante para el periodo actual.

1.3 Requisitos no funcionales:

A. **Requisito de usabilidad.** El coordinador de la maestría podrá utilizar todas las funciones del sistema después de cuatro horas de formación. Después de esta formación, el número medio de errores cometidos por usuarios experimentados no superará los dos por hora de uso del sistema.
B. **Requisito de eficiencia de desempeño.** Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos.
C. **Requisito de disponibilidad**. El sistema estará disponible durante el horario laboral normal (de lunes a viernes, de 08.00 a 18:00).
D. **Requisito de capacidad de recuperación:** La información almacenada debe guardarse en una base de datos de respaldo, cada vez que exista un cambio en el estado de la base de datos.
E. **Requisitos de seguridad:**
   - Todas las comunicaciones externas entre servidores de datos, aplicación y cliente del sistema deben estar cifradas utilizando el algoritmo RSA.
   - La información de los estudiantes se encuentra en un servidor denominado Afrodita, y la información de los docentes se encuentra en un servidor denominado Atenea. Cuando un estudiante desea iniciar sesión debe hacerlo contra el servidor Afrodita, el cual le retorna un token de seguridad. Cuando un docente desea iniciar sesión debe hacerlo contra el servidor Atenea, el cual le retorna un token de seguridad.

F. Requisitos de interoperabilidad:
   - El sistema debe consultar al sistema SIMCA para obtener los datos de un estudiante a partir de su código. El sistema SIMCA ofrece un servicio web mediante el formato JSON y el modelo REST.
   - El sistema debe consultar al sistema de recursos humanos para obtener los datos de un docente a partir de su número de identificación y tipo de identificación. El sistema de recursos humanos ofrece un servicio web mediante el formato XML y el protocolo SOAP.
   - Para enviar un correo electrónico, se debe utilizar un sistema microsoft exchange email system que existen en la Universidad.

G. **Requisitos de mantenibilidad:**
- La aplicación desarrollada debe tener un puntaje de mantenibilidad de A.
- Las tecnologías a utilizar en el desarrollo del producto software deben ser manejadas por la división de sistemas de la Universidad del Cauca
- Es necesario almacenar los eventos en una base de datos en particular. Un evento puede ser un error al ejecutar una funcionalidad o un intento de acceso no autorizado.
- Las tecnologías de la base de datos pueden cambiar con el tiempo. Por eso, es necesario que la lógica de negocio, esté aislada del medio de persistencia.

1.4 Tecnologías a utilizar:

- Aplicación web: Angular
- APP: Desarrollo nativo con Android.
- BackEnd: Spring Boot, Java, Microsoft exchange email system
- Base de datos: Motor Oracle y MySQL
- Despliegue: Amazon Web Services(AWS)
- La gestión de los datos se realiza con repositorios mediante Hibernate y JPA.

1.5 Patrones arquitectónicos mínimos

Al realizar un análisis de los requisitos funcionales y no funcionales, el diseñador a identificado que debe utilizar los siguientes patrones arquitectónicos:

Patrón Capas:. La capa es una unidad lógica, que establece como organizar el código. Por ejemplo: presentación (vista), controlador, negocio, acceso a datos. Una capa es un módulo lógico de software con su propia lógica central y límites. El patrón plantea que un sistema se puede estructurar en capas, en donde cada capa provea servicios a la siguiente capa

Patrón Niveles: El tier (nivel) es una unidad física, donde se ejecuta el código / proceso. Un nivel es un contenedor físico de una o más capas. El patrón plantea que en un nivel pueden ejecutarse varias capas.

MVC: El patrón plantea que en un sistema pueden existir las siguientes partes: Modelo(Donde se evidencien las funcionalidades y datos), Vista(Donde se muestra la información al usuario) y el Control(Donde se recibe la información del usuario)-

Cliente-servidor: Se debe reflejar las partes cliente y servidor, donde el cliente(s) solicite servicios a la parte servidor y el servidor provee servicios al cliente.

1.6 Patrones de diseño mínimos:

Inyección de dependencias: Evidenciar el uso del patrón ID en un componente.

Fachada: Reflejar el uso del patrón Facade donde se evidencie su uso para desacoplar parte de sus componentes con los clientes.

Patrón repositorio: evidenciar el uso del patrón para abstraer las operaciones sobre nuestros modelos, con la intención de ocultar la implementación de esas operaciones a las capas superiores.

2. **Entregables**

En este taller se debe trabajar en equipos de 5 personas en la etapa de diseño del sistema software solicitado.

Deben tomar como entradas los requisitos funcionales, requisitos no funcionales, tecnologías, y patrones arquitectónicos y de diseño. Posteriormente, deben realizar las actividades de: Diseño de la arquitectura, Diseño de las interfaces y Diseño de los componentes. La salida de las actividades debe ser: Especificación de la arquitectura, Especificación de los componentes y Especificación de las interfaces.

La salida debe representarse mediante 4 diagramas: diagrama de contexto, diagrama de contenedores, diagrama de componentes y diagrama de despliegue.

El diagrama de contexto debe responder las siguientes preguntas:

¿Cuál es el sistema de software que estamos construyendo?
¿Qué usuarios lo usan?
¿Con qué sistemas existentes debemos interactuar (internos a la organización y externos a la organización)?

El diagrama de contenedores debe responder a las siguientes preguntas:

¿Cuál es la forma general del sistema de software?
¿Cómo se distribuyen las responsabilidades en el sistema?
¿Cómo se comunican los contenedores entre sí?
Como desarrollador, ¿dónde necesito escribir código para implementar funciones?
¿Cuáles son las tecnologías generales que debo utilizar en cada contenedor?

El diagrama de componentes debe responder a las siguientes preguntas:

¿Cómo un contenedor está constituido de una serie de componentes?
¿Cuáles son las responsabilidades de cada componente?
¿Cuáles son los detalles de tecnología / implementación de cada componente?

El diagrama de despliegue debe responder a las siguientes preguntas:

¿En qué infraestructura tecnológica los sistemas de software y / o contenedores se despliegan?.

La infraestructura puede ser física, virtualizada (por ejemplo, IaaS, PaaS, una máquina virtual), en contenedores (por ejemplo Docker), un entorno de ejecución (por ejemplo, un servidor de base de datos, servidor web / de aplicaciones). Las tecnologías se pueden anidar. También puede incluir nodos de infraestructura como servicios DNS, balanceadores de carga, cortafuegos, etc

3. Formato

Los diagramas deben entregarse en un documento que tenga portada, tabla de contenido, introducción, diagramas, referencias. En formato .pdf. El documento debe ser subido al classroom en el link indicado, hasta el día 14 de octubre de 2022, hasta las 24:00, y será presentado al docente en un horario acordado.

Nota:

¿Cuándo crear un controlador? Un controlador es el encargado de recibir la notificación de una acción solicitada por un usuario. Por ejemplo si en un sistema se tienen recursos como estudiantes, libros etc. Debe tenerse un controlador que gestione las peticiones para estudiantes y otro controlador que gestione las peticiones relacionadas con libros. En otras palabras, debe existir una separación de responsabilidades entre los controladores. Los controladores deben usar una capa de servicios (Fachada), la cual es responsable de realizar la mayoría de cálculos. A su vez la fachada utiliza los repositorios para conectarse a la base de datos.

¿Cuándo crear un repositorio? En el contexto de persistencia en el mundo de Java, un repositorio (Repository) es una clase que se encarga de gestionar todas las operaciones de persistencia contra una tabla de una base de datos.

En el diagrama de componentes, se debe seleccionar el contenedor que ofrece los servicios a las aplicaciones relacionadas con la presentación.

El puntaje de mantenibilidad establece 5 rangos A, B, C, D y E. El rango A va del puntaje 0 al 0.05, el B va desde 0.06 al 0.1, el C va desde 0.11 al 0.20, el D va desde 0.21 al 0.5 y el E va desde el 0.51 al 1. El puntaje de mantenibilidad A quiere decir que el esfuerzo para mantener el código es menor o igual al 5% del tiempo invertido en el desarrollo de la aplicación, el puntaje de mantenibilidad B es del 6% a 10% , el puntaje C es del 11% a 20%, el puntaje D es del 21% al 50% y el puntaje E mayor al 50%. Por ejemplo, si un módulo tomó 100h/hombre, desarrollarlo y su puntaje de mantenibilidad es A, quiere decir que el costo o esfuerzo para mantener el código es menor o igual a 5h/hombre.

Si el puntaje de mantenibilidad es A quiere decir que el software desarrollado viola pocas buenas prácticas de programación, la deuda técnica es baja y por ende la capacidad de mantenimiento es alta, mientras que un puntaje de mantenibilidad E quiere decir que el software desarrollado viola muchas buenas prácticas, la deuda técnica es alta y por ende la capacidad de mantenimiento es baja. Para calcular el puntaje de mantenibilidad proponemos utilizar la herramienta SonarQube la cual se encuentra en la siguiente guía oficial https://docs.sonarqube.org/latest/.