Universidad Rey Juan Carlos – Diseño y Arquitectura Software (DAS)

**Práctica1: Captura y Representación de Decisiones de Diseño**

Curso 2020-2021

GRUPO 4 (GRUPO DE TEST)

Azahara Andújar Muñoz-Quirós

Carlota Menéndez del Campo

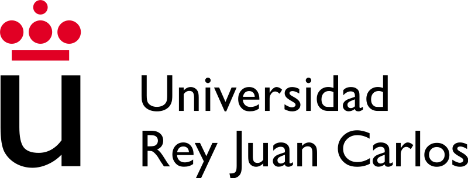
Javier Méndez García-Brioles

Alberto Jiménez Gómez

David Mestanza Rubia

Manuel Martín Aláez

**Contacto**: [a.andujar.2017@alumnos.urjc.es](mailto:a.andujar.2017@alumnos.urjc.es) (Representante)



## **Tarea 1- Análisis de requisitos:**

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Nombre** | **Descripción** |
| **RF-1** | Adaptación del producto al cliente | La presentación de las páginas web y los productos deberán adaptarse automáticamente al cliente desde PC, Tablet o móvil. De esta forma habrá que tener en cuenta que hay que hacer la migración pensada para la universalidad de clientes. |
| **RF-2** | Componentes de presentación | Son los responsables del control de la interfaz de usuario y el consumo de servicios remotos. Este componente está directamente relacionado con el RF-1 ya que será el que controlará la forma en la que se adaptan las páginas webs y los productos a los distintos clientes. |
| **RF-3** | Lógica de dominio o de negocios | La aplicación deberá tener una forma definidas unas reglas para determinar como la información que proviene del mundo real puede ser creada, almacenada y cambiada. |
| **RF-4** | Lógica de acceso a base de datos | Son los componentes de acceso a datos responsables de obtener acceso a las bases de datos (SQL o NoSQL). |
| **RF-5** | Lógica de integración de aplicaciones | Habrá que desarrollar un canal de mensajería entre aplicaciones y microservicios para que todos sepan que hacen el resto siempre que deban saberlo. Principalmente en agentes de mensajería. |
| **RF-6** | Integrar microservicios asíncronos | La aplicación deberá poder integrar microservicios de forma asíncrona para reforzar la fiabilidad de los microservicios nativos de la tienda virtual. Estos microservicios no deberán afectar a otros subsistemas |
| **RF-7** | Integrar aplicaciones externas asíncronas | La aplicación deberá poder integrar aplicaciones externas de forma asíncrona para reforzar la fiabilidad de los microservicios nativos de la tienda virtual. |
| **RF-8** | Soporte de tareas | El sistema deberá soportar las tareas de identificación de usuarios (que necesita un nombre de usuario y una contraseña), catálogo de microservicios y procesado de pedidos (que consta de varias opciones de forma de pago junto con el número del pedido), que utilizarán una base de datos SQL. |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Nombre** | **Descripción** |
| **RF-9** | Comunicación de microservicios | Los microservicios independientes se comunicarán a través del protocolo REST y de forma asíncrona. |
| **RF-10** | Cesta de compra | Habrá una función cesta de compra que usará una caché para almacenar su información dinámica (como los productos y la cantidad de dichos productos) y accederá a la localización de microservicios de terceros a través de una base de datos MangoDB. |
| **RF-11** | Bases de datos de microservicios | Cada microservicio tiene su propia base de datos, lo que permite separarlo totalmente de otros microservicios. |
| **RF-11.1** | Coherencia entre Bases de Datos | La coherencia entre las bases de datos de los diferentes microservicios se logra mediante eventos de integración de nivel de aplicación (a través de un bus de eventos lógicos), como por ejemplo mediante Command and Query Responsibility Segregation (CQRS). |
| **RF-12** | Comunicación entre clientes y microservicios | Los clientes móviles y de páginas Web se comunican con los puntos de conexión de puerta de enlace de API única enrutando las solicitudes de los clientes a los microservicios y, a continuación, se comunican con los microservicios. |
| **RF-12.1** | Comunicación entre clientes web y microservicio | Los clientes web tradicionales se comunican con el microservicio MVC. Mas adelante, El microservicio MVC se comunica con otros microservicios mediante una puerta de enlace API. |
| **RF-13** | Utilización de contenedores | El sistema constará de contenedores, dentro de los cuales se implementarán los diferentes microservicios. Los contenedores podrán ir implementados dentro de un mismo y único host o bien se podrá contar con un clúster para poder orquestar los contenedores, donde cada contenedor se ejecuta en una host diferente. En este caso cada nodo podría ejecutar un cierto número de contenedores. |
| **RF-14** | Comunicación de cliente a HTTP | Los clientes podrán comunicarse con el microservicio de HTTP a través de un Gateway que contiene diversas APIs con el fin de poder consultar las actualizaciones desde las aplicaciones cliente. |

|  |  |  |
| --- | --- | --- |
| **Identificador** | **Nombre** | **Descripción** |
| **RF-15** | Comunicación asíncrona basada en eventos. | Se podrá utilizar una comunicación asíncrona realizada a través de un bus de eventos. Se implementará bien a través de un agente de mensajería (como RabbitMQ) o de un Service Bus (como Azure Service Bus). |
| **RF-15.1** | Propagación de actualizaciones entre los microservicios. | La comunicación asíncrona basada en eventos servirá para propagar actualizaciones entre los microservicios. |
| **RF-15.2** | Integración con aplicaciones externas. | La comunicación asíncrona basada en eventos servirá para integrar aplicaciones externas. |
| **RF-16** | Comunicación con microservicios. | Las aplicaciones cliente pueden comunicarse con esos microservicios. La comunicación se hará por contenedores a través de las direcciones URL públicas publicadas por las puertas de enlace de API. |