|  |  |
| --- | --- |
| **CAPTURA Y REPRESENTACIÓN DE DECISIONES DE DISEÑO**  PRÁCTICA 1 |  |
|  |  |
| 22/11/2021  **GRUPO 8**  Hugo Coto G. ([h.coto.2018@alumnos.urjc.es](mailto:h.coto.2018@alumnos.urjc.es))|  Alberto García S. | Rodrigo Marqués B. |  Enrique Martín S. | Iván Sánchez L. |  |

# **Índice**

### Índice general

|  |  |  |
| --- | --- | --- |
|  | | |
|  | *[Aquí va el índice]* |  |
|  | | |

## 1. Roles

Arquitectos Software Senior (ASS):

* Rodrigo Marqués Buil
* Enrique Martín Santorcaz

Arquitectos Software Junior (ASJ):

* Iván Sánchez Labrador

Arquitectos Software Cognitivos (ASC):

* Hugo Coto González
* Alberto García Sroda

## 2. Introducción

El objetivo de esta práctica es realizar el diseño de una tienda virtual basada en una arquitectura Web de tres capas y se desea migrar a una arquitectura de microservicios.

Semana a semana desarrollaremos los objetivos propuestos buscando un desarrollo incremental en nuestro trabajo, que nos de pie a poner en práctica de manera ordenada los puntos que se expondrán a continuación.

Se pondrán en práctica las nociones de diseño aprendidas durante el desarrollo de la asignatura, que darán como resultado un diseño capaz de dar respuesta a todos los requisitos funcionales capturados de las peticiones y descripción del cliente de lo que espera que sea el producto a recibir.

Será en estos términos que tomaremos las decisiones de diseño en cuánto a patrones y arquitecturas que consideremos apropiadas dado el ámbito y las condiciones del proyecto. Dichas decisiones serán recogidas y documentadas en el debido formato de cara a una mejor legibilidad y estructuración de las mismas.

Cada decisión tomada tendrá su reflejo en el correspondiente diagrama que conformará el marco general de nuestro diseño, y de este modo, a través de la interrelación entre las arquitecturas y los patrones seleccionamos obtendremos una visión clara de nuestra solución.

## 3. Resultado Iteraciones

### Tarea 1: Semana 1

A continuación, se dispone la tabla con los requisitos funcionales que han sido extraídos del enunciado de la práctica en la Tarea 1: Semana 1:

|  |  |  |
| --- | --- | --- |
| ID | Nombre | Descripción |
| RF1 | Elección de arquitectura | Se indicará la elección de la arquitectura a tomar para el cliente. |
| RF1.1 | Arquitectura de microservicios | Se toma la Arquitectura basada en Microservicios como elección en el proyecto. |
| RF1.2 | Migración de la antigua arquitectura a la nueva | Se migrará todos los módulos de la aplicación para que estén basados en una arquitectura por microservicios. |
| RF1.3 | Mejorar la flexibilidad y escalabilidad | Realizar una optimización en la bases de datos y realizar buenas praxis en la utilización de la arquitectura. |
| RF2 | Componentes de presentación | Responsables de la parte cliente. |
| RF2.1 | Control de interfaz de usuario | Se indicará como se controla la interfaz de usuario. |
| RF2.2 | Consumo de servicios remotos | Se indicará como controlar el consumo de servicios remotos. |
| RF3 | Componentes lógica de negocio | Responsable de la funcionalidad de la aplicación. |
| RF3.1 | Módulo de pedidos y compras | Se indicará cómo realizar las compras y pedidos para usuario. |
| RF3.2 | Módulo detección de preferencias | Se indicará la información necesaria y trata correcta de la información de preferencias. |
| RF3.3 | Módulo de conexión a sistemas de pago | Se indicará de que forma se realizarán los pagos y con que medios. |
| RF3.4 | Módulo de mensajería a dispositivos móviles | Se indicará la forma de la mensajería a dispositivos móviles. |
| RF3.5 | Módulo de seguridad en compras | Se indicará la seguridad en la que se basa la arquitectura. |
| RF3.6 | Módulo de devoluciones | Se indicará como se realizaran las devoluciones para clientes. |
| RF4 | Lógica de acceso a bases de datos | Son los componentes responsables para obtener acceso a las BBDD. |
| RF5 | Sistema de mensajería por middelware independiente | Implementación de middleware único para mensajería. |
| RF6 | Integración asíncrona de los microservicios | Se implementará para soportar un mayor numero de compras y preferencias. |
| RF7 | Creación nueva base datos | Se creará para almacenar la localización de los microservicios. |
| RF8 | Integración bus eventos lógicos | Para una correcta coherencia de los datos entre las distintas BBDD. |
| RF9 | Determinar número de contenedores | Limitación de intentos de compra a 5 intentos. |
| RF10 | Conexión y microservicios | Definición de tipo de conexión a la aplicación y entre microservicios. |
| RF10.1 | Definir vía de conexión | Conexión a la aplicación. |
| RF10.2 | Definir comunicación y monitorización de los microservicios | Conexión y monitorización de los microservicios. |

*(Tabla de requisitos)*

**Tarea 2: Semana 2**

En esta segunda semana, empezamos a tomar las decisiones de diseño en función a los requisitos funcionales que habíamos extraído la semana anterior. Primeramente, comenzamos por decidir qué tipo de arquitectura iba a ser la principal.

La arquitectura principal de este producto, es una arquitectura de microservicios, ya que es la demanda que el cliente ha solicitado expresamente para mejorar la flexibilidady escalabilidad del sistema que se quiere migrar desde una arquitectura Web de tres capas.

|  |  |
| --- | --- |
| ***Título de la decisión*** | Arquitectura de microservicios |
| ***ID*** | ADD-0001 |
| ***Fecha*** | 30-10-2021 (último update) |
| ***Creadores*** | Enrique Martín S. & Rodrigo Marqués Buil |
| *ASC* | Hugo Coto G. |
| ***Estado*** | Aceptada |
| ***Requisitos (directrices de decisión)*** | RF1: “Elección de arquitectura”  RF1.1: “Arquitectura de microservicios” |
| ***Decisiones alternativas (opciones)*** | Opción 1(única): Arquitectura de microservicios |
| ***Decisión final tomada (opción seleccionada)*** | Opción 1(única): Arquitectura de microservicios |
| ***Consecuencias positivas*** | * Opción 1:   + **Es el que ha solicitado el cliente**   + Mejoras en la flexibilidad   + Mejoras en la escalabilidad   + Menos problemas para el cliente   + Código y mantenimiento más legible   + Facilidad a la hora de aplicar una implementación continua |
| ***Consecuencias negativas*** | * Opción 1:   + Posible dificultad a la hora de migrar el sistema   + Tener verificado en la base de datos todos los microservicios del sistema para un mayor control |
| ***Documentación*** | https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices |

*(Tabla ADR-0001)*

Aunque esta arquitectura también tiene sus inconvenientes, como pueden ser el gran consumo de recursos que implica, lidiar con la complejidad adicional de los sistemas distribuidos o implementar dependencias de un servicio a otro, la hemos elegido, en primer lugar, porque es la arquitectura que quiere el cliente, y además, facilitan el despliegue, el mantenimiento y el testeo de los microservicios.

|  |  |
| --- | --- |
| ***Título de la decisión*** | Migración a nueva arquitectura |
| ***ID*** | ADD-0001 |
| ***Fecha*** | 30-10-2021 | 14-11-2021(revisión) |
| ***Creadores*** | Enrique Martín S. & Rodrigo Marqués Buil |
| *ASC* | *- No hay ASC en esta decisión -* |
| ***Estado*** | Aceptada |
| ***Requisitos (directrices de decisión)*** | RF1.2: “Migración de la nueva arquitectura”  RF1.3: “Mejora de flexibilidad y escalabilidad” |
| ***Decisiones alternativas (opciones)*** | --- |
| ***Decisión final tomada (opción seleccionada)*** | --- |
| ***Consecuencias positivas*** | * Seguridad en implementación de escalabilidad * Seguridad en implementación de flexibilidad * Revisión continua de estas características * Nivel detallado de estas características |
| ***Consecuencias negativas*** | * No se vislumbra una escalabilidad palpable desde el comienzo del desarrollo de la migración * Posible pérdida del contacto con estos requisitos durante el desarrollo de la migración |
| ***Documentación*** | https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices |

*(Tabla ADR-0001 (1))*

|  |  |
| --- | --- |
| ***Título de la decisión*** | Gestión de compras y acceso a plataforma de pagos |
| ***ID*** | ADD-0004 |
| ***Fecha*** | 4-11-2021 (último update) |
| ***Creadores*** | Enrique Martín S. & Rodrigo Marqués Buil |
| *ASC* | Alberto García S. |
| ***Estado*** | Aceptada |
| ***Requisitos (directrices de decisión)*** | RF3.2: “Módulo de detección de preferencias”  RF3.3: “Módulo de sistema de pago” |
| ***Decisiones alternativas (opciones)*** | * **Opción 1:** la BBDD almacenará las preferencias de los usuarios en base a las búsquedas anteriormente realizadas en la aplicación, debe estar implementada mediante un solo microservicio con dos API desarrolladas en ***JAVA***. * **Opción 2:** La lógica es la misma que en la opción 1, pero las API desarrolladas en ***Phyton*** puesto que es un lenguaje más potente e intuitivo. * **Opción 3:** La lógica es la misma que en la opción 1, pero las API desarrolladas en ***Node*** ya que es una opción perfectamente válida. |
| ***Decisión final tomada (opción seleccionada)*** | Opción 2 |
| ***Consecuencias positivas*** | * **Comunes:**   + El sistema de preferencias generado por búsquedas anteriores permite que el usuario encuentre productos parecidos a los que le interesan, y por tanto, que sea una herramienta útil para el cliente.   + La BBDD NoSQL estaría correctamente conectada con los diferentes módulos del sistema, y recopilar la información de los diferentes usuarios no es una operación complicada de realizar ni implementar, por lo que la escalabilidad se vuelve a respetar en este punto.   + Actualizar la base de datos una vez el usuario ha dejado de navegar por la aplicación con las nuevas preferencia generadas, permitirá que el dicho sistema de preferencias sea flexible y se adapte a los nuevos gustos de los usuarios.   + La forma de conectar con el banco es fácil de implementar puesto que muchas aplicaciones anteriormente han utilizado esta lógica, por lo que podemos encontrar mucha documentación y ejemplos en diferentes webs. * **Opción 1:**   + JAVA es un lenguaje preparado para albergar este tipo de aplicaciones, lo que puede ser de ayuda a la hora de implementarlo y escalarlo. * **Opción 2:**    + La potencia nativa de Python sumado con lo intuitivo que es el lenguaje, lo convierte en una opción más que factible. * **Opción 3:**    + Podemos expandir nuestro sistema añadiendo módulos de forma fácil gracias al NPM\* además de la alta escalabilidad que permite el uso de clusters. |
| ***Consecuencias negativas*** | * Separar en un sistema de dos APIs las implementaciones propuestas podría resultar confuso a la hora de conectarlas con los módulos de usuario e interfaz. * La lógica no parece revisable. * La base de datos NoSQL puede verse afectada por una sobrecarga de importancia en la lógica planteada. |
| ***Documentación*** | https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices |

*(Tabla ADR-0004)*