DOCUMENTO DE ENTREGA DE MICROSERVICOS: MODULO CONTRATISTAS

Portal

Fábrica de Desarrollo de Software

**REVISION Y CONTROL DE CAMBIOS**

**Revisión y Versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Autor | Versión | Descripción | Cargo | Fecha |
| Carlos Torres | 1.0 | Documento | Desarrollador | 19/02/2024 |
|  |  |  |  |  |

**Control de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Autor | Versión | Descripción del cambio |
| 19/02/2024 | Carlos Torres | 1.0 | Liberación Documento |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# TABLA DE CONTENIDO

[TABLA DE CONTENIDO 4](#_Toc159247250)

[RESUMEN 5](#_Toc159247251)

[INTRODUCCION 5](#_Toc159247252)

[PROPOSITO GENERAL DEL SISTEMA O SOLUCION 5](#_Toc159247253)

[ALCANCE DEL SISTEMA 5](#_Toc159247254)

[OBJETIVOS 5](#_Toc159247255)

[RESTRICIONES Y LIMITACIONES 6](#_Toc159247256)

[DEFINICIONES Y ABREVIATURAS 6](#_Toc159247257)

[REQUISITOS 7](#_Toc159247258)

[Requisitos Funcionales 7](#_Toc159247259)

[Requisitos Suplementarios (No funcionales) 7](#_Toc159247260)

[DEFINICIÓN DE LA ARQUITECTURA 8](#_Toc159247261)

[Modelo de Referencia 8](#_Toc159247262)

[ENTREGA DE LA SOLUCION 9](#_Toc159247263)

[Detalles del proyecto 9](#_Toc159247264)

[Equipo desarrollo 9](#_Toc159247265)

[Detalles del software 9](#_Toc159247266)

[Contenido de la entrega 9](#_Toc159247267)

[Requerimientos del sistema 9](#_Toc159247268)

[Listado Microservicios 10](#_Toc159247269)

[Listado Utilidades 10](#_Toc159247270)

[Microservicios 10](#_Toc159247271)

[EcosystemEureka 11](#_Toc159247272)

# RESUMEN

El presente documento describe los componentes implementados para el módulo contratistas del portal. También se comparte la arquitectura de software utilizada y la estrategia de implementación guías y recomendaciones para la correcta operación.

# INTRODUCCION

Se entiende por arquitectura del software, al conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del software en tiempo de ejecución. Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto. Este documento describe en términos generales, las ideas principales que definen la solución implementada.

# PROPOSITO GENERAL DEL SISTEMA O SOLUCION

El propósito de este documento es proporcionar información general, clara y concisa de la solución implementada.

# ALCANCE DEL SISTEMA

Contratistas es un módulo que se utiliza para gestionar los procesos de solicitudes de pagos con los cuales se liquidan los contratos que una entidad adquiere con un tercero.

# OBJETIVOS

A continuación, se describen los objetivos de acuerdo a los requerimientos entregados al área de fábrica.

* Definir la arquitectura soportada en el sistema.
* Implementar la infraestructura que soportará la solución.
* Generar componentes/librerías Core.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 formuario presentar cuentaDefinir lineamientos para la creación del modelo de datos que soporta el sistema.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 listado procesos en atención.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 aprobar-rechazar solicitud supervisor.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 listado de liquidación masiva contador.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 formulario visualizar para liquidar.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 consulta de solicitudes de pago.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 consulta procesos revisados.
* F104\_ING\_HU\_Y\_Criterios\_De\_Aceptacion\_V1.1 anulación de liquidaciones.

# RESTRICIONES Y LIMITACIONES

* No se implementó el componente Gateway por problemas en Cors desde el front Angular.
* No se implementó Api de seguridad para proteger las url desde el backend,
* Algunas integraciones se realizan a través de la base de datos para soportar compatibilidades.
* Pendiente implementación de herramientas de monitoreo de microservicios y el servicio de mensajería Kafka.
* Existen dependencias con el módulo de Viáticos.

# DEFINICIONES Y ABREVIATURAS

|  |  |
| --- | --- |
| Termino | Definición |
| Springboot | Spring Boot es un framework de código abierto para Java que facilita y acelera el desarrollo de aplicaciones web y microservicios. |
| Docker | Docker es una plataforma de código abierto que facilita el desarrollo, la implementación y la ejecución de aplicaciones mediante contenedores. |
| Contenedor | Un contenedor es un paquete ligero, portátil y autosuficiente que incluye todo lo necesario para ejecutar una aplicación: código, runtime, herramientas del sistema, bibliotecas y configuraciones. Los contenedores se aíslan unos de otros y del sistema operativo host, lo que los hace ideales para ejecutar aplicaciones en diferentes entornos. |
| API | Interfaz de programación de aplicaciones |
| Microservicio | Los microservicios son un estilo arquitectónico para el desarrollo de software que implica la construcción de aplicaciones como una colección de servicios pequeños, independientes y desplegables individualmente. Estos servicios se comunican entre sí a través de APIs bien definidas, generalmente utilizando protocolos ligeros como REST o gRPC. |

# REQUISITOS

Los requisitos definidos en la arquitectura, no son otros sino los requisitos y necesidades, tanto de nuestros clientes, las normativas de ley y las mejoras que surgen de nuestra mejora constante para ofrecer un mejor producto que satisfaga las necesidades de nuestros actuales y futuros usuarios del sistema.

En esta sección se indican los requisitos atendidos por el diseño o arquitectura que se describen a continuación:

## Requisitos Funcionales

A continuación, se describen los casos de uso relevantes de la arquitectura del sistema:

* El Sistema de controlar el acceso de los usuarios
* El Sistema debe visualizar las opciones por roles.
* El Sistema debe permitir la gestión de múltiples clientes con un solo despliegue.
* El Sistema debe generar pagos sin bloqueos recurrentes.

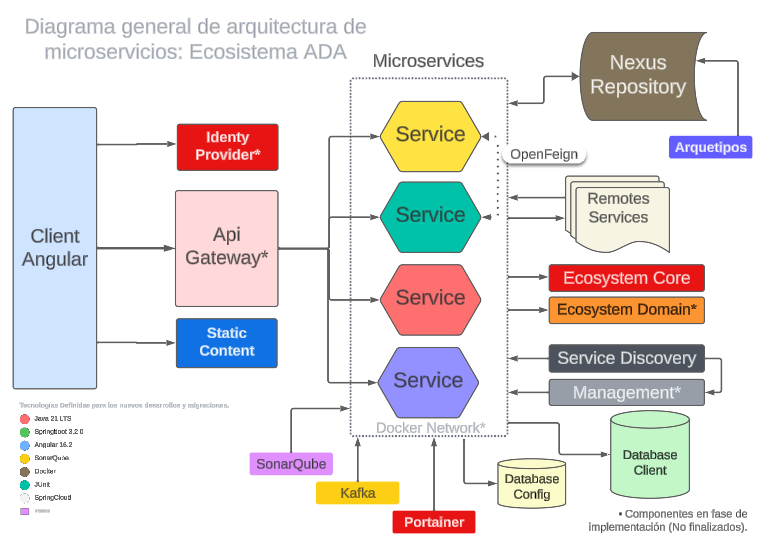
## Requisitos Suplementarios (No funcionales)

* Usabilidad
* Eficiencia
* Concurrencia
* Seguridad
* Interoperatividad, Confiabilidad, Escalabilidad
* Facilidad de Prueba
* Reusabilidad
* Facilidad de Instalación
* Disponibilidad

# DEFINICIÓN DE LA ARQUITECTURA

## Modelo de Referencia

A continuación, se describe el modelo de referencia utilizado en la solución:



|  |  |  |
| --- | --- | --- |
| Componente | Descripción | Estado |
| Client Angular | Front de la aplicación (Vistas) | Implementado |
| Identy Provider | Proveedor de identidad (seguridad) | Parcial |
| Api Gateway | Servicio de enrutamiento y descubrimiento | Pendiente |
| Static Content | Contenido estático (imágenes, reportes, plantillas) | Implementado |
| SonarQube | Servicio de verificación de estándares de desarrollo | Parcial |
| Kafka | Servicio de mensajería | Implementado |
| Portainer | Aplicación de gestión de contenedores | Implementado |
| Docker Network | Red virtual para la comunicación de los contenedores Docker. | Implementado |
| Service | Microservicios que contienen la solución | Implementado |
| OpenFeign | Interfaz de comunicación entre microservicios | Implementado |
| Nexus Repository | Repositorio de gestión de dependencias | Implementado |
| Remotes Services | Servicios remotos de consumos | Pendiente |
| Ecosystem Core | Librería base utilizada para la construcción de microservicios | Implementado |
| Ecosystem Domain | Librería general que contiene dtos del modelo de datos de la solución. | Implementado |
| Arquetipos | Plantilla base de microservicios | Implementado |
| Service Discovery | Servicio de registro de microservicios | Implementado |
| Management | Servicio de monitoreo | Pendiente |
| Database Client | Bases de datos de los clientes | Implementado |
| Database Config | Base de datos de configuración. |  |

# ENTREGA DE LA SOLUCION

## Detalles del proyecto

|  |  |
| --- | --- |
| Nombre del Proyecto | Contratistas |
| ****Cliente/Equipo Receptor**** | QA, Implantación, Soporte y Mantenimiento, Líder de Producto |

## Equipo desarrollo

|  |  |  |
| --- | --- | --- |
| Backend | Carlos Alberto Torres Camargo | [carlos.torres@ada.co](mailto:carlos.torres@ada.co) |
| Frontend | Luis Ardila | [luis.ardila@ada.co](mailto:luis.ardila@ada.co) |
| Requirement analyst/Test analyst | Paola Ayala | [paola.ayala@ada.co](mailto:paola.ayala@ada.co) |

## Detalles del software

|  |  |
| --- | --- |
| Nombre del Software | Portal: Contratistas |
| Versión Global | 1.0.0 |
| Descripción | Gestiona las solicitudes de pagos realizadas por los contratistas. |

## Contenido de la entrega

A continuación, se listan los ítems que hacen parte de la entrega:

* Requerimientos del sistema
* Listado de Microservicios
* Listado de Utilidades
* Listado de Librerías

### Requerimientos del sistema

La solución ha sido implementada en una arquitectura de microservicios. A continuación, se listan las necesidades de hardware y software del sistema.

La solución ha sido implementada en una arquitectura de microservicios. A continuación, se listan las necesidades de hardware y software del sistema.

#### Ambiente Productivo

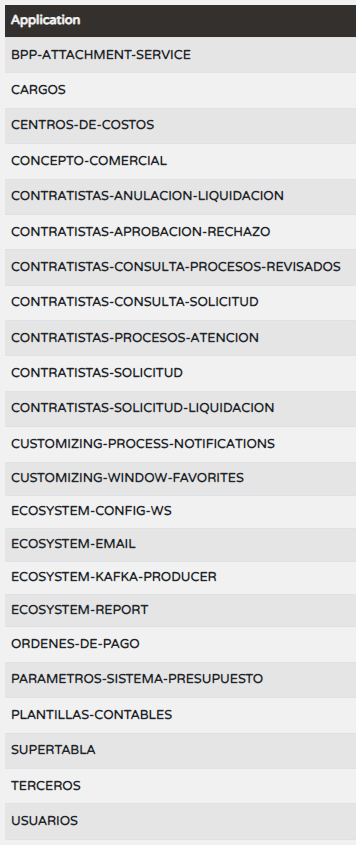
|  |  |
| --- | --- |
| Sistema Operativo | Ubuntu Server 20.04 LTS |
| Memoria RAM | 64GB |
| HDD | 256GB |
| Docker | version 24.0.7, build afdd53b |
| Docker Compose | version v2.23.1 |
| kafka | Versión 3.5.1 |
| Java JDK | Amazon Corretto 21.0.1.12 |

#### Ambiente Desarrollo, QA, Presentación

|  |  |
| --- | --- |
| Sistema Operativo | Ubuntu Server 20.04 LTS |
| Memoria RAM | 32GB |
| HDD | 256GB |
| Docker | version 24.0.7, build afdd53b |
| Docker Compose | version v2.23.1 |
| kafka | Versión 3.5.1 |
| Java JDK | Amazon Corretto 21.0.1.12 |

### Listado Microservicios

La solución ha requerido los siguientes microservicios:



### Listado Utilidades

La solución ha requerido utilidades que ayuden a gestionar los microservicios. A continuación, se listan las utilidades utilizadas:

* portainer/portainer-ce:latest: Herramienta para gestionar contenedores.
* EcosystemTemplateProject: Plantilla generadora de microservicios.

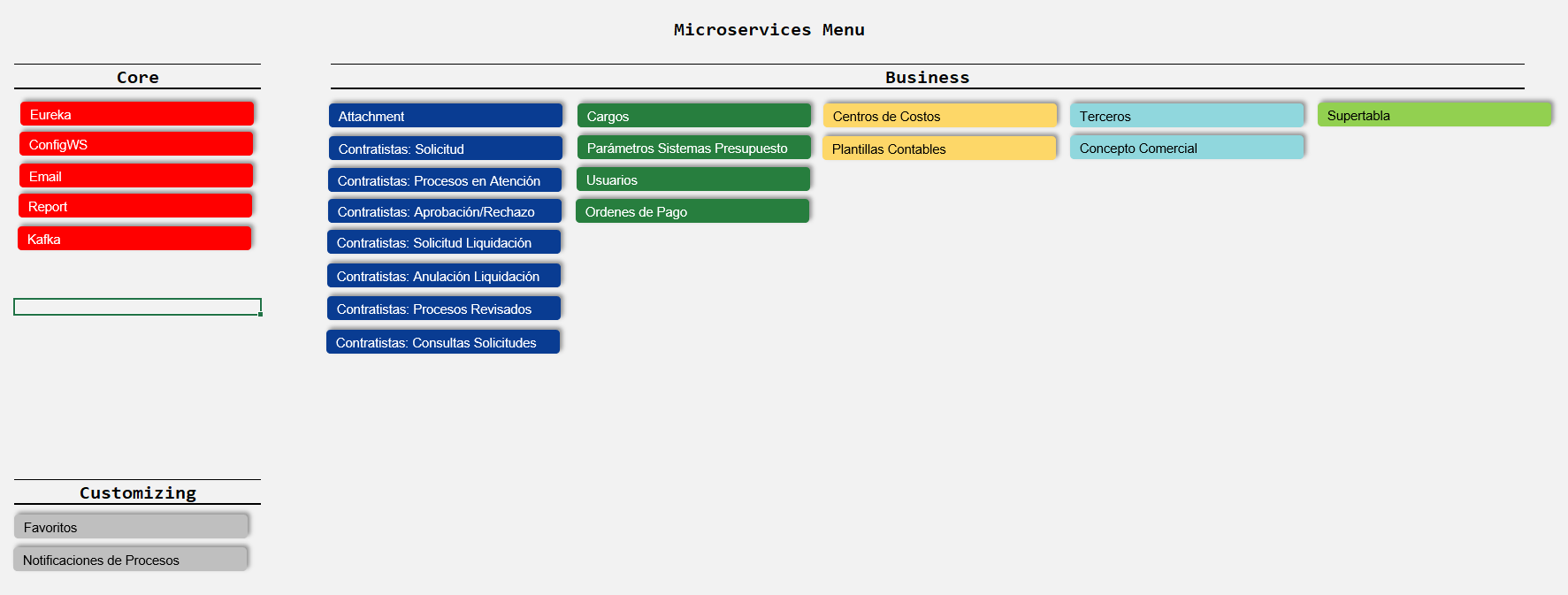
### Listado Librerías

La solución ha requerido la implementación de librerías que soportan la solución:

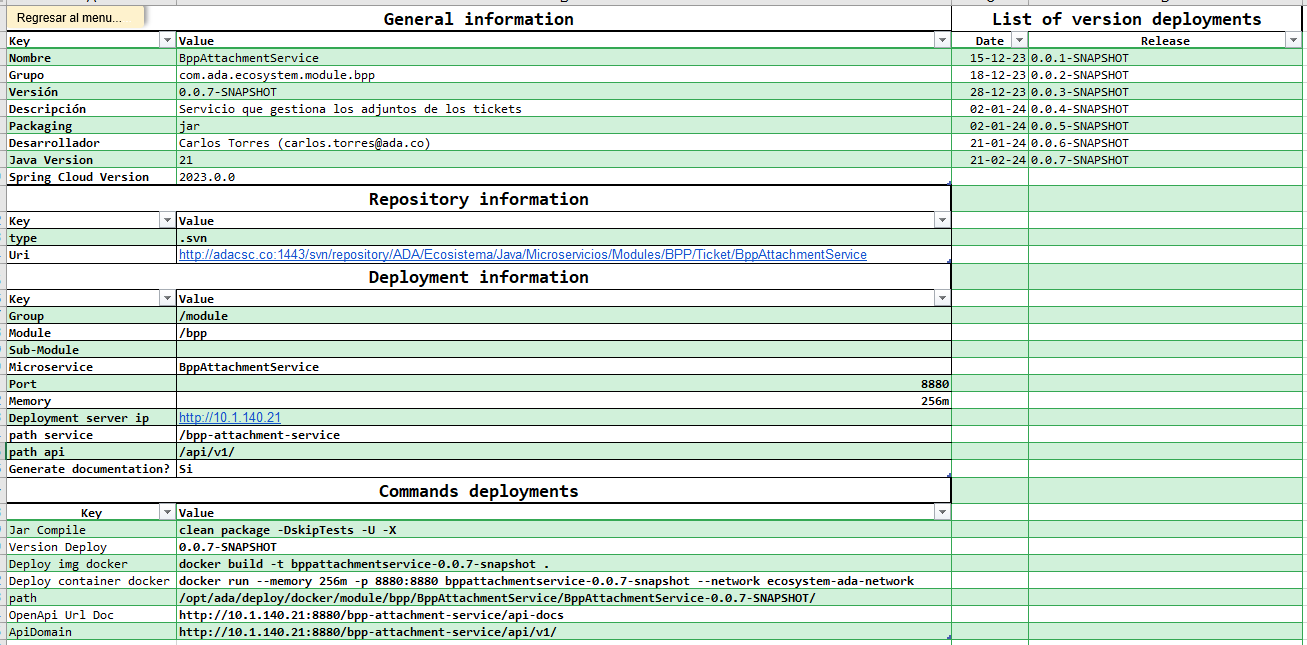
* EcosystemCore: Clases Core de la solución.
* EcosystemDomain: Modelo del dominio del negocio.

### Microservicios

El listado de microservicios requiere una serie de pasos para el proceso de despliegue. Se implementa una hoja de cálculo en la cual se administra esa información y se generan los comandos automáticamente para evitar errores de despliegue. La herramienta cuenta con un menú de control como el que se visualiza en la siguiente imagen:



Cada opción dirige a una hoja de cálculo en la cual se muestra la información del microservicio.



La información de la hoja se divide en los siguientes segmentos:

* **General information**: Información general del proyecto java del microservicio, es extraída del pom.
* **Repository information**: Información del repositorio donde se aloja el proyecto.
* **Deployment information**: Información de despliegue, contiene la ubicación del artefacto de despliegue y el path base del servicio.
* **List of version deployments:** En esta sección se registran las versiones liberadas.
* **Commands deployments:** En esta sección se generan los comandos necesarios para generar los artefactos necesarios para desplegar el microservicio. A continuación, se describen los comandos:
  + **Jar Compile:** Comando para generar el jar para generarlo se debe contar con un cliente Maven el cual se provee automáticamente desde el IDE Eclipse, SpringTools.}
  + **Version Deploy:** En este campo se despliega la lista de versiones registradas en la sección List of versión deployments. Se debe selecciona una versión para generar los comandos.
  + **Deploy img docker:** Comando para generar la imagen Docker. Se debe en el servidor de despliegue por medio de un cliente ssh.
  + **Deploy container docker:** Comando para generar y ejecutar el contenedor docker que contiene la imagen docker.
  + **Path:** Ruta del servidor donde se debe alojar el artefacto.
  + **OpenApi Url Doc:** Url por defecto donde se publica la información del microservicio.
  + **ApiDomain:** Url base del microservicio.

Adicionalmente se pueden incluir otros comandas específicos del microservicio según se requiera para el funcionamiento.

* Deploy Ecosystem Ada.xlsx
* Ecosystem.postman\_collection.json Colección de pruebas backend.