**[MultiRental Storage]**

**(SAD) Software Architecture Document**

**Versión 1.0**

**Identificación de Documento**

| **Identificación** |  |
| --- | --- |
| **Proyecto** |  |
| **Versión** |  |

| **Documento mantenido por** |  |
| --- | --- |
| **Fecha de ultima revisión** |  |
| **Fecha de próxima revisión** |  |

| **Documento aprobado por** |  |
| --- | --- |
| **Fecha de última aprobación** |  |

**Historia de Revisiones**

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**1**](#_heading=h.30j0zll) **Introducción 3**

[1.1](#_heading=h.1fob9te) Contexto del Problema 3

[1.2](#_heading=h.3znysh7) Propósito 3

[1.3](#_heading=h.2et92p0) Ámbito 3

[1.4](#_heading=h.tyjcwt) Definiciones, acrónimos y abreviaciones 3

[1.5](#_heading=h.3dy6vkm) Referencias 3

[1.6](#_heading=h.1t3h5sf) Resumen ejecutivo 4

[**2**](#_heading=h.2bn6wsx) **Representación de la Arquitectura 5**

[2.1](#_heading=h.4d34og8) Representación 5

[**3**](#_heading=h.2s8eyo1) **Metas y Restricciones de la Arquitectura 6**

[A continuación se revisan las metas y restricciones de la arquitectura. 6](#_heading=h.17dp8vu)

[3.1](#_heading=h.3rdcrjn) Metas de la arquitectura 6

[3.2](#_heading=h.26in1rg) Restricciones de la Arquitectura 6

[3.3](#_heading=h.lnxbz9) Otros antecedentes y consideraciones 6

[**4**](#_heading=h.35nkun2) **Vista de Casos de Uso y Escenarios de Calidad 8**

[4.1](#_heading=h.1ksv4uv) Modelo de Casos de Uso 8

[4.2](#_heading=h.44sinio) Especificación de Casos de Uso Relevantes 8

[4.3](#_heading=h.2jxsxqh) Especificación de los Escenarios de Calidad Relevantes 9

[**Vista Lógica 11**](#_heading=h.z337ya)

[4.4](#_heading=h.3j2qqm3) Parte Estructural 11

[4.5](#_heading=h.1y810tw) Parte Dinámica 11

[**5**](#_heading=h.4i7ojhp) **Vista de Procesos 13**

[**6**](#_heading=h.2xcytpi) **Vista de Implementación 14**

[**7**](#_heading=h.1ci93xb) **Vista de Despliegue 15**

[**8**](#_heading=h.3whwml4) **Decisiones de Diseño y Selección de Alternativas 16**

1. **Introducción**
   1. **Contexto del Problema**

Sodimac, una de las principales cadenas de tiendas de mejoramiento del hogar en Chile, enfrenta dificultades en la gestión de inventarios de herramientas para arriendo. Actualmente, la falta de un sistema centralizado genera problemas en la disponibilidad de herramientas, retrasos en la actualización de inventarios y una experiencia insatisfactoria para los clientes.

* 1. **Propósito**

Este documento tiene como objetivo definir la arquitectura del sistema APT (Arriendo de Productos y Herramientas de Sodimac), que optimiza la gestión de inventarios y el arriendo de herramientas a través de una plataforma centralizada. El sistema buscará automatizar el proceso de actualización del stock en las sucursales y mejorar la eficiencia operativa.

* 1. **Ámbito**

El proyecto abarca el desarrollo e implementación de un sistema de gestión de inventario utilizando Flask para el frontend, proporcionando una interfaz ligera y rápida, y MariaDB para el backend, ofreciendo una base de datos en tiempo real, escalable y fácil de integrar. El sistema estará disponible en todas las sucursales de Sodimac en la Región Metropolitana de Chile.

* 1. **Definiciones, acrónimos y abreviaciones**

| **ACRONIMO** | **DESCRIPCION** |
| --- | --- |
| *APT* | Arriendo de Productos y Herramientas |
| *API* | Interfaz de Programación de Aplicaciones |
| *DB* | Base de datos |
| *Flask* | Framework web ligero de Python |
| *Firebase* | Plataforma de desarrollo de aplicaciones con base de datos en tiempo real |
| *UX* | Experiencia del usuario |

* 1. **Referencias**

A continuación se listan las referencias a otros documentos:

* **Casos de Uso** 
  1. **Resumen ejecutivo**

Este documento describe la arquitectura del sistema APT, diseñado para centralizar la gestión de inventarios de herramientas en Sodimac. El sistema utiliza **Flask** para el frontend, facilitando la interacción de los empleados con una interfaz ligera y eficiente, y **MariaDB** para el backend, proporcionando una base de datos escalable en tiempo real que optimiza la disponibilidad del stock y la experiencia del cliente.

* 1. **Representación**

La arquitectura del sistema APT sigue el enfoque del **framework 4+1** y las recomendaciones del proceso unificado. Las vistas incluidas en este documento son:

* **Vista de Casos de Uso y Escenarios de Calidad**: Presenta los casos de uso más importantes, incluyendo a los actores clave y sus interacciones con el sistema. También detalla los escenarios de calidad relacionados con el rendimiento y la disponibilidad.
* **Vista de Metas y Restricciones**: Describe las restricciones tecnológicas, normativas y estándares que influyen en las decisiones arquitectónicas, particularmente en la integración con **Flask** y **MariaDB**.
* **Vista Lógica**: Detalla los módulos principales del sistema, sus responsabilidades y cómo interactúan. Incluye el tipo de vista "Módulos" para representar la estructura lógica y "Componentes y Conectores" para el comportamiento dinámico entre **Flask** y **MariaDB**.
* **Vista de Procesos**: Explica cómo los procesos del sistema se coordinan, detallando las relaciones de comunicación entre el frontend (**Flask**) y el backend (**MariaDB**) en tiempo real.

1. **Metas y Restricciones de la Arquitectura**

A continuación se revisan las metas y restricciones de la arquitectura.

* 1. **Metas de la arquitectura**

De acuerdo con las reuniones y el análisis de los requerimientos, las principales metas arquitectónicas del sistema APT son las siguientes:

* **Desempeño**: El sistema debe ser capaz de manejar un alto volumen de consultas de inventario en tiempo real desde diversas sucursales sin afectar la velocidad de respuesta.
* **Tolerancia a fallos**: El sistema debe estar diseñado para recuperarse de posibles fallos de comunicación entre el frontend y el backend, garantizando la integridad de los datos.
* **Seguridad**: Dado que el sistema gestionará información sensible relacionada con inventarios y arriendos, debe garantizarse que las conexiones entre el frontend y el backend estén protegidas con protocolos de seguridad adecuados.
* **Modificabilidad**: La arquitectura debe permitir la fácil modificación y expansión del sistema, especialmente para integrar nuevas sucursales o funcionalidades en el futuro.
* **Operatividad**: El sistema debe ser fácil de operar por parte del personal de Sodimac, ofreciendo una interfaz clara y accesible.
  1. **Restricciones de la Arquitectura**

A continuación, se detallan las principales restricciones que pueden influir en las decisiones de diseño del sistema:

* **Tiempo de construcción:** El desarrollo del sistema cuenta con un plazo de 16 semanas, lo que obliga a priorizar funcionalidades clave y garantizar entregas continuas durante el desarrollo.
* **Infraestructura**: Se utilizarán servicios en la nube, específicamente MariaDB, lo que elimina la necesidad de servidores replicados o balanceadores de carga físicos, pero también depende de la disponibilidad de la red de Google.
* **Otros componentes de software**: No se considerará la adquisición o uso de software de terceros con licencias costosas, priorizando herramientas gratuitas o de código abierto.
  1. **Otros antecedentes y consideraciones**

La arquitectura del sistema se basa en herramientas y tecnologías que permiten un desarrollo ágil y escalable. Se consideran los siguientes componentes clave para satisfacer las metas arquitectónicas:

* Framework Flask: para el frontend, se seleccionó Flask debido a su ligereza y flexibilidad, permitiendo una rápida implementación de la interfaz de usuario y su integración con el backend de Firebase.
* Firebase para la base de datos y autenticación: MySQL permitirá una gestión en tiempo real del inventario, además de contar con una infraestructura robusta que ofrece escalabilidad y servicios de autenticación.
* Seguridad: El sistema usará protocolos como HTTPS para garantizar la seguridad en las transacciones de datos entre el frontend y el backend.

1. **Vista de Casos de Uso y Escenarios de Calidad**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

* 1. **Modelo de Casos de Uso**

El modelo de casos de uso puede ser encontrado en el documento “Casos de Uso”.

* 1. **Especificación de Casos de Uso Relevantes**

Los casos de uso considerados los más relevantes para el desarrollo de la arquitectura fueron determinados. Los criterios usados para dicha determinación fueron:

* Su implementación implica varios nodos de la vista de despliegue.
* Su implementación es de alto riesgo.
* Incluye muchos conceptos y relaciones del dominio.
* Incluye posibles escenarios críticos de calidad.

A continuación se listan los casos de uso relevantes, los cuales pueden ser encontrados con su especificación detallada en el documento “Casos de Uso”.

| **Código** | **Nombre** | **Actores** | **Prioridad** |
| --- | --- | --- | --- |
| CU-001-001 | Consultar disponibilidad de herramientas | Empleados de Sodimac | Alta |
| CU-002-001 | Registrar herramienta | Empleados de Sodimac | Muy alta |
| CU-002-002 | Eliminar herramienta arrendada | Empleado de Sodimac | Alta |
| CU-002-003 | Actualizar herramienta | Empleado |  |
| CU-002-004 | Actualizar stock de herramientas | Sistema de inventarios | Alta |
| CU-002-005 | Buscar y filtrar herramientas |  |  |
| CU-003-001 | Visualizar stock en local |  |  |
| CU-003-002 | Visualizar stock en otras sucursales |  |  |
| CU-003-003 | Visualización del estado y cantidad de herramientas |  |  |
| CU-004-001 | Generar reporte de herramientas arrendadas | Sistema | Media |
| CU-004-002 | Generar reportes de estado de herramientas |  |  |
| CU-004-003 | Generar reportes en XLSX y CSV |  |  |
| CU-004-004 | Generar reportes de inventario |  |  |
| CU-005-001 | Crear usuarios |  |  |
| CU-005-002 | Gestionar roles de usuario |  |  |
| CU-005-003 | Asignar roles y permisos |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. **Especificación de los Escenarios de Calidad Relevantes**

Después de un análisis en conjunto con los stakeholders, los escenarios de calidad se expresan a continuación:

**ID: QS1**

**Nombre:** Desempeño: Tiempo de respuesta en la consulta de inventario.

**Sinopsis**: El sistema debe permitir consultar el inventario disponible en cada sucursal sin demoras perceptibles para el usuario.

**Entorno**:Operación normal del sistema, donde los empleados de Sodimac consultar el stock de herramientas..

**Cambio en el entorno**:Alta demanda de consultas simultáneas en horarios pico.

**Comportamiento esperado**:El sistema debe responder las consultas de inventario en menos de 1 segundo por solicitud, incluso bajo carga alta.

**Medida**:El tiempo de respuesta para cada consulta de inventario debe ser menor a 1 segundo bajo condiciones normales..

**Prioridad Arquitectónica**: Alta

**Aplicación**: Global

**ID: QS2**

**Nombre:** Tolerancia a fallos:Recuperación ante un fallo en la base de datos.

**Sinopsis**:El sistema debe poder recuperarse de un fallo en la conexión con la base de datos MariaDB sin interrumpir completamente el servicio.

**Entorno**: La base de datos MariaDB operando normalmente y proporcionando datos para las consultas y registros de arriendos.

**Cambio en el entorno**: Falla momentánea en la conexión con la base de datos o tiempo de inactividad planificado.

**Comportamiento esperado**:El sistema debe ser capaz de notificar al usuario sobre el fallo, reintentar la conexión automáticamente y continuar el proceso en menos de 5 minutos después de restablecer la conexión

**Medida**: El sistema debe restablecer la conexión con la base de datos y reanudar su funcionalidad en menos de 5 minutos.

**Prioridad Arquitectónica**: Alta

**Aplicación**: Local (Aplicable en cada sucursal)

**Vista Lógica**

A continuación se presenta una vista lógica de la aplicación expresado en dos diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (módulos), y otra vista que representa la parte dinámica (componentes y conectores).

* 1. **Parte Estructural**

En el siguiente diagrama de Módulos se observa que el principal módulo….

**Ilustración 1: Diagrama de Módulos**

* 1. **Parte Dinámica**

La parte dinámica….

**Ilustración 2: Diagrama de Componentes y Conectores**

1. **Vista de Procesos**

A continuación se muestra una vista de procesos, en la cual se observa que:

**Ilustración 3: Diagrama de Procesos**

1. **Vista de Implementación**

En esta vista se aprecia que existirán dos módulos principales que contendrán distintas funcionalidades de la aplicación. A continuación se describen:

**Ilustración 4: Vista de Implementación**

1. **Vista de Despliegue**

En esta vista se despliegan los nodos que participan con el sistema. Los nodos principales son los nodos Servidor de Integración. Características a continuación:

**Ilustración 6: Diagrama de Despliegue**

1. **Decisiones de Diseño y Selección de Alternativas**

Las principales decisiones arqutectónicas se tomaron en consideración de la restricción **Tiempo de Construcción**. Dado que el proyecto debe implementarse en un tiempo ajustado y sin holguras, se privilegió la adopción de una arquitectura conocida y que presente un bajo riesgo en su implementación.

Asimismo, la arquitectura se modularizó con el primer objetivo de separar concernimientos de forma que permita paralelización en construcción de dichos componentes, y que a su vez sea módulos testeables unitariamente de forma de asegurar (mediante la suite Junit) que cada pieza tenga una baja tasa de fallas.

Un segundo elemento fue considerado en la arquitectura, que corresponde a la restricción de **Infraestructura** con que debe cumplir la aplicación, combinado con el escenario de calidad de **Tolerancia a Fallos**, nos condiciona la modularización de la aplicación en una **aplicación web activa-activa** y una **aplicación de servicios activa-pasiva**.

El escenario de calidad relacionado con la **mantenibilidad** nos conduce al modelamiento pensando en la separación de concernimiento de los componentes y a la utilización del patrón **provider** de forma que el sistema pueda delegar sus requerimientos de información hacia sistemas externos a piezas de software no acopladas que nos permitan su extensibilidad a futuro.

Por último, se eligió que la estrategia para implementar los providers externos en aquellos servicios asíncronos de entrada se implementarían mediante un temporizador (quartz) que levantará los procesos que verifican la llegada de información a las colas de entrada (mensajería asíncrona, archivos de texto en directorios). Esta estrategia fue seleccionada para disminuir el riesgo pues es una solución simple y efectiva.

1. **Análisis de Reutilización**

[Poner razonamiento de componentes que se reutilizarán o que se desarrollarán]