

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto A*PI y Funciones de Lugares***

Curso: *Tópicos de Base de Datos Avanzados*

Docente: *Patrick Jose Cuadros Quiroga*

Alumno:

***Mayner Gonzalo Anahua Coaquira (2020067145)***

**Tacna – Perú**

***2024***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | MPV | ELV | ARV | 10/10/2020 | Versión Original |

Sistema API y Funciones de Lugares

Documento de Arquitectura de Software

Versión *1.0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | MPV | ELV | ARV | 10/10/2020 | Versión Original |

INDICE GENERAL

Contenido

[1. INTRODUCCIÓN 5](#_Toc69808834)

[1.1. Propósito (Diagrama 4+1) 5](#_Toc69808835)

[1.2. Alcance 5](#_Toc69808836)

[1.3. Definición, siglas y abreviaturas 5](#_Toc69808837)

[1.4. Organización del documento 5](#_Toc69808838)

[2. OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS 5](#_Toc69808839)

[2.1.1. Requerimientos Funcionales 5](#_Toc69808840)

[2.1.2. Requerimientos No Funcionales – Atributos de Calidad 5](#_Toc69808841)

[3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA 6](#_Toc69808842)

[3.1. Vista de Caso de uso 6](#_Toc69808843)

[3.1.1. Diagramas de Casos de uso 6](#_Toc69808844)

[3.2. Vista Lógica 6](#_Toc69808845)

[3.2.1. Diagrama de Subsistemas (paquetes) 7](#_Toc69808846)

[3.2.2. Diagrama de Secuencia (vista de diseño) 7](#_Toc69808847)

[3.2.3. Diagrama de Colaboración (vista de diseño) 7](#_Toc69808848)

[3.2.4. Diagrama de Objetos 7](#_Toc69808849)

[3.2.5. Diagrama de Clases 7](#_Toc69808850)

[3.2.6. Diagrama de Base de datos (relacional o no relacional) 7](#_Toc69808851)

[3.3. Vista de Implementación (vista de desarrollo) 7](#_Toc69808852)

[3.3.1. Diagrama de arquitectura software (paquetes) 7](#_Toc69808853)

[3.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes) 7](#_Toc69808854)

[3.4. Vista de procesos 7](#_Toc69808855)

[3.4.1. Diagrama de Procesos del sistema (diagrama de actividad) 8](#_Toc69808856)

[3.5. Vista de Despliegue (vista física) 8](#_Toc69808857)

[3.5.1. Diagrama de despliegue 8](#_Toc69808858)

[4. ATRIBUTOS DE CALIDAD DEL SOFTWARE 8](#_Toc69808859)

[Escenario de Funcionalidad 8](#_Toc69808860)

[Escenario de Usabilidad 8](#_Toc69808861)

[Escenario de confiabilidad 9](#_Toc69808862)

[Escenario de rendimiento 9](#_Toc69808863)

[Escenario de mantenibilidad 9](#_Toc69808864)

[Otros Escenarios 9](#_Toc69808865)

**1. INTRODUCCIÓN**

**1.1 Propósito (Diagrama 4+1)**

El propósito del documento es presentar la arquitectura del sistema para el proyecto API de Lugares, utilizando el modelo 4+1. Este sistema está diseñado para gestionar lugares, direcciones y categorías mediante una API RESTful. El enfoque está en satisfacer tanto los requisitos funcionales como no funcionales, priorizando la eficiencia y la portabilidad del sistema.

**1.2 Alcance**

El documento se centra en la vista lógica del framework y sus aspectos fundamentales. Se incluyen diagramas de arquitectura y procesos, así como los detalles de implementación y despliegue.

**1.3 Definición, siglas y abreviaturas**

* API: Interfaz de Programación de Aplicaciones.
* CRUD: Operaciones de Crear, Leer, Actualizar y Eliminar.
* FastAPI: Framework web para el desarrollo de APIs rápidas y escalables.
* CouchDB: Base de datos NoSQL orientada a documentos.
* Docker: Herramienta para la contenedorización de aplicaciones.

**1.4 Organización del documento**

El documento está dividido en secciones que cubren:

* Introducción al proyecto y objetivos arquitectónicos.
* Representación de casos de uso y diseño lógico.
* Atributos de calidad del software y escenarios clave.

**2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS**

**2.1 Priorización de Requerimientos**

**2.1.1 Requerimientos Funcionales**

|  |  |  |
| --- | --- | --- |
| ID | Descripción | Prioridad |
| RF1 | Gestión de lugares (CRUD). | Alta |
| RF2 | Gestión de direcciones (CRUD). | Alta |
| RF3 | Gestión de categorías (CRUD). | Media |
| RF4 | Documentación automática mediante Swagger. | Media |

**2.1.2 Requerimientos No Funcionales – Atributos de Calidad**

|  |  |  |
| --- | --- | --- |
| ID | Descripción | Prioridad |
| RNF1 | Seguridad mediante autenticación y autorización. | Alta |
| RNF2 | Escalabilidad para manejar grandes volúmenes de datos. | Alta |
| RNF3 | Disponibilidad del 99.9%. | Alta |
| RNF4 | Respuesta en menos de 200 ms para consultas simples. | Media |

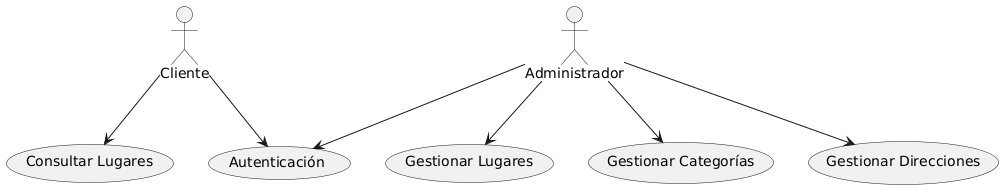
**2.2 Restricciones**

* Uso exclusivo de CouchDB como base de datos.
* Implementación mediante FastAPI.
* Despliegue obligatorio con Docker.

**3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA**

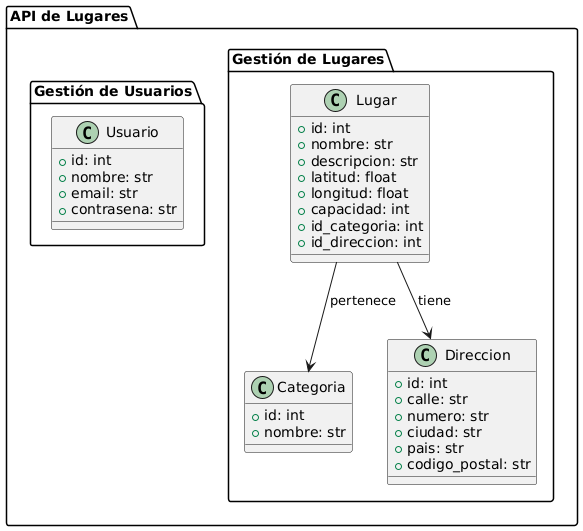
**3.1 Vista de Caso de Uso**

**3.1.1 Diagramas de Casos de Uso**



**3.2 Vista Lógica**

**3.2.1 Diagrama de Subsistemas (Paquetes)**



**3.2.2 Diagrama de Secuencia**

Diagrama

Descripción generada automáticamente

**3.2.5 Diagrama de Clases**

Diagrama

Descripción generada automáticamente

**4. ATRIBUTOS DE CALIDAD DEL SOFTWARE**

|  |  |
| --- | --- |
| Escenario | Descripción |
| Funcionalidad | El sistema proporciona las operaciones CRUD completas necesarias para gestionar recursos como lugares, direcciones y categorías. Además, incluye documentación automática generada con Swagger. |
| Usabilidad | La interfaz de la API está diseñada para ser intuitiva y accesible para desarrolladores, con documentación integrada que incluye Swagger y ReDoc para facilitar el aprendizaje y uso. |
| Confiabilidad | El sistema implementa mecanismos de autenticación y autorización para proteger los datos de acceso no autorizado. La disponibilidad está asegurada con un SLA del 99.9%. |
| Rendimiento | La API garantiza tiempos de respuesta inferiores a 200 ms para consultas simples, y el diseño asegura escalabilidad para manejar operaciones más complejas de manera eficiente. |
| Mantenibilidad | La utilización de contenedores Docker permite una implementación uniforme en diferentes entornos, simplificando las tareas de mantenimiento y asegurando la portabilidad del sistema. |