**Documento de Arquitectura de Software**

SISTEMA DE MONITOREO DE EVENTOS

**Versión <1.0>**

NOTA: Este template tiene algunos textos pre cargados que deberán borrarse o revisarse y ajustarse de ser necesario. Los mismos están a modo de ejemplificar lo que se busca indicar en alguna de las secciones. Así mismo las secciones pueden eliminarse o agregar las que se crean necesarias. El objetivo del mismo es transmitir las decisiones de diseño tomadas, los caminos analizados, los pros y contras del diseño, y la propuesta de diseño final

**Historia de Revisión**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 21-04-2018 | 1.0 | Primera versión del Documento de Arquitectura para la aplicación Sistema de Monitoreo de Eventos |  |

**Tabla de Contenidos**

[**Introducción**](#_rr49e2udmdy2) **4**

[Propósito](#_1fob9te) 4

[Alcance](#_b3qsjq11e2me) 4

[Definiciones, Acrónimos y Abreviaturas](#_18lbx9u3ei30) 4

[Referencias](#_315kb9g1oc4k) 4

[Visión General](#_3kr5k7f5lxz) 4

[**Representación Arquitectural**](#_4d34og8) **5**

[**Metas de Arquitectura y Restricciones**](#_j9cfy3kjfj4r) **5**

[**Análisis del problema**](#_e5m0762gumt3) **6**

[**Decisiones Tomadas**](#_o4e0uvlwaadt) **6**

[**Vista de Casos de Uso**](#_8epe9iaspo4j) **6**

[….](#_rzogehh5iyzc) 6

[….](#_sge99ck0my4g) 6

[**Vista Lógica**](#_l7572l9qv0o7) **6**

[**Vista de Componentes (Física)**](#_emu4kndxdu3h) **6**

[**Vista de Despliegue**](#_tter6faq51g8) **7**

[**Vista de Procesos**](#_1y810tw) **7**

[**Tamaño, Rendimiento, Memoria**](#_q17wricyi7oa) **7**

[**Calidad**](#_2wps7wu48k45) **7**

[**Manejo de Errores**](#_bjuq4hu0gtfs) **7**

[Buenas Prácticas](#_p494rhcllvcj) 8

[Política del Manejo de Excepciones](#_55hqmpewq1w4) 8

**Documento de Arquitectura**

# **Introducción**

## **Propósito**

El presente documento provee una visión inicial para la arquitectura de la aplicación “Sistema de Monitoreo de Eventos”, a través de la utilización de las 4+1 vistas de Kruchten. De esta manera, se busca capturar y asentar las decisiones importantes que serán tomadas en el desarrollo de la aplicación.

## **Alcance**

Se muestra a alto nivel el diseño de la arquitectura por vistas de la aplicación. En cada una, se presentan los diagramas correspondientes, a saber: modelo conceptual, diagrama de clases, casos de uso, diagramas de interacción, entre otros.

## **Definiciones, Acrónimos y Abreviaturas**

1. *Evento*

Cualquier dato generado por un sistema externo que posee información de un suceso.

1. *…*

## **Referencias**

Se hace referencia a los documento de material de clases del repositorio de la materia Técnicas de Diseño de FIUBA: https://github.com/7510-tecnicas-de-disenio/material-clases.

## **Visión General**

El presente documento se estructura de la siguiente manera: representación arquitectural, metas y restricciones de la arquitectura, presentación de las 4+1 vistas, tamaño y desempeño del software y finalmente la calidad del sistema.

# **Representación Arquitectural**

Para esta entrega del documento, se representarán las vistas utilizando los siguientes recursos:

* Vista de Casos de Uso.

A cada uno se le hará una descripción en formato breve para enunciar su Escenario Principal de Éxito. Se utilizará el Diagrama de Casos de Uso en notación UML.

* Vista Lógica.

Se realizará el Modelo Conceptual de la aplicación, que permita comprender el dominio del problema, utilizando la notación UML. Además, se desarrollará la primera versión del Diagrama de Clases de la aplicación para representar el dominio de la solución, utilizando también la notación UML y patrones.

* Vista de Componentes.

Se explicará la estructura que describe el modelo de implementación de la aplicación, su composición en capas y cada uno de sus componentes.

* Vista de Despliegue

Se muestra la relación de la aplicación a desarrollar con el hardware requerido para el despliegue del sistema.

* Vista de Procesos.

Se habla de los procesos (si son varios y amerita describirlos).

# **Metas de Arquitectura y Restricciones**

El desarrollo de la aplicación se enfoca en que el sistema sea escalable y soporte recibir y procesar gran cantidad de eventos, aplicando las reglas.

Nuestras principales metas a nivel de arquitectura son las siguientes:

* Performance: El desempeño de la aplicación debe ser eficiente para ser capaz de procesar gran cantidad de eventos.
* Esacalabilidad: Se desea que el sistema pueda escalar a medida que la cantidad de eventos lo requiera
* Usabilidad: El diseño debe ser orientado por y para la comodidad del usuario, de manera que la interfaz sea intuitiva y fácil de manejar, al mismo tiempo que se fomente altamente la interacción entre ambos. De la misma forma, el usuario debe tener la capacidad de equivocarse y regresar a un estado seguro en el que se le permita cumplir con su objetivo original sin que se le haga tedioso o complicado el proceso para llegar a dicho fin.

# Análisis del problema

Se analiza el problema, y lo que se determina es que se puede separar en distintos subsistemas….

# Decisiones Tomadas

Queremos dejar detalladas las decisiones tomadas, describiendo las alternativas analizadas y sus pros y contras observadas:

….

# **Vista de Casos de Uso**

En esta vista se mostrarán la lista de los Casos de Usos críticos de la aplicación “Sistema de Monitoreo de Eventos”

## ….

….

## 

….

## ….

….

# **Vista Lógica**

Como propuesta, se tiene el siguiente Modelo Conceptual para cuáles son las asociaciones pertinentes entre los conceptos más relevantes para la aplicación.



# **Vista de** Componentes **(F**í**sica)**

# **Vista de Despliegue**

A continuación se muestra un diagrama de despliegue que modela, a alto nivel, la distribución de las piezas de software de la aplicación sobre los elementos de hardware que se usarán para ejecutarla, indicando las asociaciones entre nodos con caminos de comunicación, que indican la tecnología requerida para que ésta se lleve a cabo exitosamente.

...

En el diagrama se observa que ….

# **Vista de Procesos**

…..

# **Tamaño**, **Rendimiento**, Memoria

La aplicación …..

# **Calidad**

La arquitectura del software se ha diseñado para ser independiente de la plataforma en la que el sistema se utilice. Ya que esto ofrece al sistema la capacidad de ser portable, la aplicación se beneficia por igual de esta portabilidad.

La aplicación estará desarrollada por capas, de manera tal que las capas más superficiales como la de interfaz no afectarán a la capa lógica. Cada capa se comunicará con las capas que estén directamente por debajo de ellas. No existirán saltos de acceso entre capas de manera tal que se mantenga una eficiente comunicación entre ellas.

# **Manejo de Errores**

En el desarrollo de un sistema es importante crear una buena política de manejo de errores de tal manera que se cubran la mayoría de los posibles casos de mal funcionamiento durante el uso de todas las funciones que ofrece el sistema. Además, la información que puede ofrecer una buena política de manejo de excepciones es muy valiosa a la hora de depurar el sistema y por lo tanto, el mantenimiento puede simplificarse a gran escala.

## **Buenas Prácticas**

Para el manejo de errores se seguirán las siguientes prácticas:

* Encapsulamiento: en el caso de que una excepción viaje a través de las capas del sistema y se cree alguna otra a partir de la primera generada, se guarda toda la información del primer error a través del encapsulamiento en el momento de la creación de la nueva excepción. De esta forma, no se pierde información debido a la propagación de la excepción o de la sustitución a través de las capas del sistema.
* Lanzar temprano: las excepciones son lanzadas en el preciso instante en el que ocurre el error. Así se tiene un mejor control a la hora de depurar y de buscar el origen de la falla.
* Atrapar tarde: las excepciones se atraparán cuando se tenga la mayor cantidad de información posible y cuando se esté en la capa correspondiente del sistema que tenga las herramientas necesarias para manejar correctamente el error.
* Obtener y desplegar la mayor cantidad de información: a la hora de generar un error, la mayor cantidad de información debe ser desplegada de tal manera que se pueda ubicar fácilmente el origen y por lo tanto, se depure rápidamente el sistema

## **Política del Manejo de Excepciones**

La política de manejo de errores en el sistema se definirá por capas. Para cada capa aplica una política distinta que permitirá que las excepciones que permitan manejar la mayor cantidad de errores del sistema en general.

**Capa de Datos y Utilidades**

En esta capa se utilizará una política de Propagación y de encapsulado de excepción. Esto tendrá como consecuencia que la información del error ocurrido no se pierda y se transmita lo más posible a través de su viaje por las capas del sistema.

**Capa Lógica**

Para la capa lógica se aplicará una política de sustitución y encapsulamiento. Se sustituye la excepción proveniente de la capa inferior de tal manera que se cree alguna otra más específica a la causa del error y que, por consiguiente, brinde información más específica a la capa superior.

**Capa de Presentación e Interfaz**

En la capa de interfaz se aplicará la política de manejo de errores. En esta capa del sistema se realizará el manejo de todas las excepciones provenientes de las capas inferiores y se desplegará al usuario la información necesaria al error ocurrido.