<Sistema Gestor SAMOC>

Arquitectura

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Propósito

Este Documento de Arquitectura de Software (SAD) proporciona una descripción de la arquitectura del Sistema Gestor SAMOC; comprendiendo los procesos importantes: Gestionar Socios, Gestionar clientes, Gestionar Aportes, entre otros procesos de la Sociedad. Haciendo uso de diversas visiones arquitectónicas para representar diversos aspectos del Sistema.

Este documento tiene como objetivo, servir de referencia para el diseño y futura implementación, brindando a los implementadores y especialistas técnicos la información necesaria para el desarrollo del sistema.

# Objetivos de la Arquitectura y filosofía

|  |  |
| --- | --- |
| **Principio/Política** | El propósito es:   * Definir una arquitectura candidata para el sistema a partir de las experiencias previas en sistemas problemas similares. * Definir patrones arquitectónicos, mecanismos de diseño y convenciones de modelado. * Definir la estrategia de reutilización |
| **Descripción** | En el documento se pretende definir las piezas y/o componentes del sistema, así como sus relaciones, organizándolas en capas bien definidas con dependencias específicas. Además del uso de patrones de diseño que proporcionen una solución común a un problema específico en un contexto dado; esta solución puede ser parcial o total. |
| **Motivación** | Utilizar un patrón arquitectónico que exprese un esquema de organización estructural fundamental para nuestro sistema de software. De este modo, proveer un conjunto predefinido de subsistemas, especificando sus responsabilidades, e incluyendo sus reglas y guías para organizar las relaciones entre sí. |
| **Implicaciones** | Con la arquitectura propuesta se pretende lograr los siguientes objetivos:   * Permitir la Conexión concurrente de varios usuarios. * La arquitectura propuesta debe evitar la inconsistencia de datos. |

# Suposiciones y Dependecias

El sistema propuesto estará implementado con la tecnología Microsoft .NET en el lenguaje de programación C# en dos entornos que son:

1. Entorno Web: Uso de ASP .NET WEB FORM.

Esto significa que el entregable tiene una gran ventaja; que es la de poder ejecutar desde cualquier navegador web (Firefox, Google Chrome, Safari, etc.).

1. Entorno Mobile: OS Android v4.4 con uso de html5, Javascript, JSON, JQuery, Kendo Mobile UI, CSS.

El producto final y/o entregable tendrá una extensión .apk que podrá ser instalada en cualquier dispositivo con Sistema Operativo Android con versión 4 a más.

# Requisitos Arquitectónicamente significativos

Entre las características que limitan nuestra elección en la construcción del nuestro sistema, tenemos las siguientes:

* **Desempeño:** En cada transacción que realice el usuario, no se debe perder la integridad de los datos ya que esto podría tener impactos significativos en el sistema.
* **Tiempo de respuesta:** el tiempo que el sistema tarde en responder a las interacciones con el usuario debe ser inmediata, tal es el caso que cada respuesta no debe demorar más de 8 segundos.
* El sistema debe estar planificado para funcionar en cambio ambiente duro, tal es el caso que no se debe producir un problema muy conocido como es el congelamiento del sistema.
* **Escalabilidad:** nuestro sistema debe estar en capacidad de permitir que en el futuro se puedan desarrollar nuevas funcionalidades, modificar o eliminar funcionalidades.
* **Facilidad de uso:** nuestro sistema debe ser de fácil uso y entendimiento por parte del usuario además de mostrar mensajes de error para alertar al usuario.
* **Seguridad:** el sistema estará preparado para restringir el acceso a usuarios no autorizados además de restringir permisos a los usuarios registrados.
* **Validaciones**: El sistema indicará al usuario que campos son obligatorios y los tipos de datos de cada campo.
* **Mantenibilidad**: esta característica será brindada gracias a la utilización de Arquitectura en Capas.
* **Interoperabilidad**: nuestro sistema estará en la capacidad de proveer información a un aplicativo móvil para, así tener informado al administrador de los casos puntuales de la empresa.

# Las decisiones, limitaciones, y justificaciones

* Restricciones de Tiempo.
* Restricciones de Seguridad y Privacidad.
* Restricciones de Tecnología.

|  |  |
| --- | --- |
| **Limitación Arquitectónica** | Tiempo |
| **Código y Nombre** | 01-Restricción de tiempo |
| **Declaración del Problema** | Existe una fecha de entrega a la cual el proyecto debe estar terminada |
| **Supuestos** | Terminar a tiempo para la entrega final |
| **Motivación** | Aprender nuevas herramientas para el desarrollo de software |
| **Alternativas** | Aprovechar el tiempo para la ejecución del proyecto |
| **Decisión** | Realizar los entregables en el tiempo establecido |
| **Estatus** | En proceso |
| **Justificación** | El manejo de mucha información en poco tiempo genera demoras |
| **Implicaciones** | Entregables a tiempo. |

|  |  |
| --- | --- |
| **Limitación Arquitectónica** | Seguridad y Privacidad |
| **Código y Nombre** | 02-Restricciones de Seguridad y Privacidad |
| **Declaración del Problema** | Prestar toda la seguridad a los usuarios. |
| **Supuestos** | Poder acceder al sistema. |
| **Motivación** | Aprender nuevas herramientas para el desarrollo de software |
| **Alternativas** | Aprovechar de la fortaleza de la seguridad para los datos del usuario |
| **Decisión** | Realizar los entregables en el tiempo establecido |
| **Estatus** | En proceso |
| **Justificación** | La protección de la seguridad de la información de los datos |
| **Implicaciones** | Entregables a tiempo. |

|  |  |
| --- | --- |
| **Limitación Arquitectónica** | Tecnología |
| **Código y Nombre** | 03-Restricción de tecnología |
| **Declaración del Problema** | Usar la tecnología que está a nuestro alcance. |
| **Supuestos** | A través de la tecnología mejorar los procesos |
| **Motivación** | Aprender nuevas herramientas para el desarrollo de software |
| **Alternativas** | Aprovechar la tecnología para la ejecución del proyecto |
| **Decisión** | Realizar los entregables en el tiempo establecido |
| **Estatus** | En proceso |
| **Justificación** | Mejorar el proceso con la tecnología que tenemos |
| **Implicaciones** | Entregables a tiempo. |

# Mecanismos de la Arquitectura

## La arquitectura está representada siguiendo las recomendaciones de OPENUP, Las vistas necesarias se enumeran a continuación.

|  |  |  |
| --- | --- | --- |
| **VISTA** | **ELEMENTO DE MODELADO** | **DESCRIPCIÓN** |
| Caso de Uso | Diagrama | Describe el proceso de negocio más significativo y el modelo del dominio. Presenta los actores y los casos de uso para el sistema. |
| Vista de Restricciones |  | Describe restricciones tecnológicas, normativas, uso de estándares, entre otros, las cuales deben ser respetadas tanto por el proceso de desarrollo como por el producto desarrollado |
| Lógica | Diagrama de Clases | Describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias. |
| Vista de Procesos |  | Describe los procesos concurrentes del sistema |
| Implementación | Diagrama | También llamada vista de desarrollo, describe la organización estática del software. |
| Modelo de Datos |  | Presenta los modelos de datos, los servicios de persistencia. |
| Implantación | Diagrama | También conocida como vista física y se encarga de modelar la disposición física del software. |

# Abstracciones Clave

Entre los elementos más importantes que influyen en la realización del sistema serán los siguientes:

* **Desempeño:** en cada transacción que realice el usuario, no se debe perder la integridad de los datos ya que esto podría tener impactos significativos en el sistema.
* **Tiempo de respuesta:** el tiempo que el sistema tarde en responder a las interacciones con el usuario debe ser inmediata, tal es el caso que cada respuesta no debe demorar más de 10 segundos.
* El sistema debe estar planificado para funcionar en cambio ambiente duro, tal es el caso que no se debe producir un problema muy conocido como es el congelamiento o cuelgue del sistema.
* **Seguridad**: el sistema estará preparado para restringir el acceso a usuarios no autorizados además de restringir permisos a los usuarios registrados.
* **Mantenibilidad**: esta característica será brindada gracias a la utilización de Arquitectura en Capas.
* **Costo:** debido al bajo presupuesto con el que cuenta el Complejo Baños del Inca, se desarrollará en Visual Studio 2012 Express y SQL Server 20008 Express, por temas de licencia.
* **Patrón Repository:** se utilizará como mediador entre el dominio de la aplicación y los datos que le dan persistencia. Con esto podemos decir que el usuario de este repositorio no necesitaría conocer la tecnología utilizada para acceder a los datos, sino que le bastaría con saber las operaciones que nos facilita este “mediador”, el repositorio.
* **Patrón Inyección de Dependencias:** no servirá para inyectar comportamientos a componentes, esto quiere decir, extraer responsabilidades a un componente para delegarlas en otro, estableciendo un mecanismo a través del cual el nuevo componente pueda ser cambiado en tiempo de ejecución.
* **Patrón Modelo Vista Vista-Modelo**: este patrón nos permite la posibilidad de reutilizar, mantener y escalar código sin ningún problema, realizar pruebas unitarias a módulos específicos de la aplicación: por ejemplo realizar una prueba unitaria al geo referencia; y poder separar los roles entre un diseñador y el desarrollador.

# Capas o marco arquitectónico

Se decidió utilizar este marco arquitectónico para ayudar a reducir el impacto ante algún cambio tecnológico y además permite el mantenimiento del producto a largo plazo.

**Arquitectura en capas**

Hoy en día es común trabajar en una aplicación en la cual su arquitectura está dividida en capas.

Por ejemplo, en una clásica aplicación Web están los típicos proyectos de “Presentación”, “Lógica de Negocio” y “Persistencia”, los que vendrían a ser capas Lógicas.

El objetivo de hacer una separación de capas es que cada una de esta cumpla un rol específico y tenga responsabilidades bien definidas.

**Capa de Presentación:** esta capa trata de los aspectos gráficos de la presentación, la cual será la encargada de interactuar con el usuario.

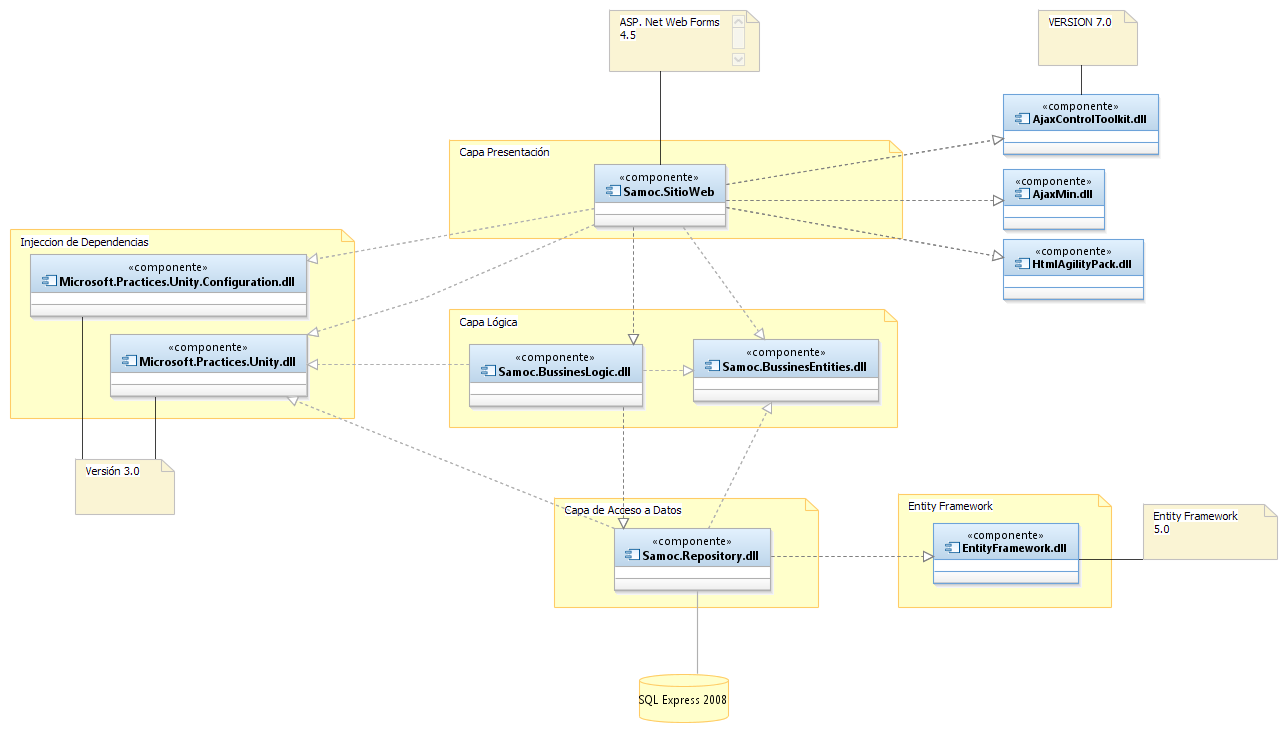
**Capa de Lógica de Negocio:** esta capa debe ocuparse de todo lo relacionado a la lógica del negocio de la aplicación y mas no a la presentación o persistencia de datos del usuario.

**Capa de Persistencia de Datos:** Esta capa se encarga de obtener y persistir los datos.

De esta forma podemos lograr encapsular funcionalidades y permitir la reusabilidad de código.

**Capa de Entidad:** esta capa trabaja con las entidades que serán mapeadas (mediante code first) para que sean las tablas de base de datos.

**Capa de Servicio:** Esta capa es la encargada de ofrecer la comunicación Rest Full (POST, GET, PUT, DELETE), es decir cliente-servidor con la aplicación móvil



# Vistas de Arquitectura

A continuación se muestran las vistas en total en la representación arquitectónica del sistema.

## Vistas Recomendadas:

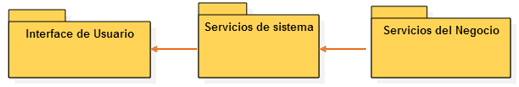
* **Vista Lógica**

Esta vista presenta tres niveles de Arquitectura; a continuación se grafican los niveles a tratar.



**Arquitectura del sistema:**

En el primer nivel se especifica el estilo de arquitectura para el sistema de gestión de información. El siguiente diagrama presenta la arquitectura del sistema.



Cada capa determina un rol para los módulos que residen en ella. La Interfaz de Usuario tiene como objetivo el manejo de la lógica del usuario. Está conformada por el conjunto de páginas web dinámicas que interactúan entre si y por módulos que rodean la lógica de los casos de uso.

Los Servicios del Sistema representan los servicios básicos que debe proveer el sistema; estos servicios son directamente utilizados por los módulos de la capa superior.

Los Servicios de Negocio son servicios de manejo de información del negocio; son servicios aún más básicos.

**Arquitectura Lógica:**

La Arquitectura Lógica presenta un refinamiento de la Arquitectura del Sistema. La dimensión Requerimientos, principalmente la Vista de Casos de uso, va a verse realizada por los módulos aquí presentados. Se analizará los módulos presentes en cada capa de la Arquitectura del Sistema, presentando finalmente la Arquitectura Lógica.

**Interface de usuario:**

Esta capa consiste de un módulo por cada caso de uso identificado. Cada módulo contiene la lógica que lleva adelante el caso de uso y un conjunto de páginas dinámicas utilizadas por dicha lógica.

**Arquitectura de Módulos:**

La Arquitectura de los Módulos presenta un refinamiento de la Arquitectura Lógica. Esta incluye, para cada módulo, una vista que mejor define su diseño. Para cada tipo de módulo, para los módulos de cada capa, se utilizará una vista diferente.

* **Vista de Casos de Uso:**

Esta vista presenta la percepción que tiene el usuario de las funcionalidades del sistema. Se presenta el proceso de negocio más importante del sistema y los casos de uso que se derivan de éste.

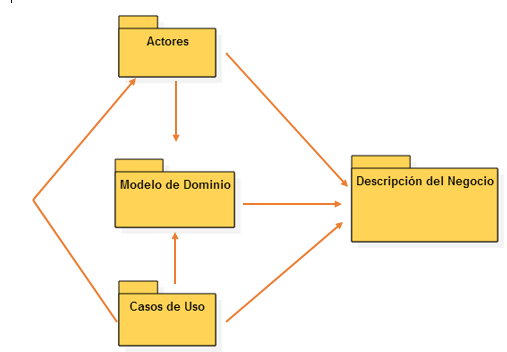


Diagrama Vista General

**DESCRIPCION DEL NEGOCIO:**

A continuación se describe el Sistema Gestor SAMOC, procesos del negocio y subprocesos que se relaciona.

**SISTEMA GESTOR SAMOC:** La Institución **“Sociedad Auxilio Mutuo Obrero – Cajamarca”** es una organización de derecho privado con personería jurídica, autónoma, apolítica, y sin fines de lucro, que promueve el desarrollo humano, cuyos miembros se rigen por el presente estatuto y su reglamento interno; cuya sigla es “SAMOC”.

El Sistema Gestor SAMOC será capaz de ayudar en las labores que se desarrollarán en relación a la Gestión de Socios y al alquiler de ambientes; este sistema contará con las funciones necesarias e importantes que requiere la Institución.

Maneja el tema de gestión de socios, se verá el registro del socio, el aporte mensual que realiza como también la entrega de beneficios mutuales a su heredero.

Sistema Gestor SAMOC trabaja con C# y .Net Framework, ambos crean el proyecto que se está desarrollando y logrando las funcionalidades de la Gestión de Socios y del Alquiler de ambientes.