

UNIVERSIDAD PRIVADA DE CIENCIAS ADMINISTRATIVAS Y TECNOLÓGICAS

RM. 409/07 – DS. 1081/11

**PROYECTO DE GRADO**

**“DESARROLLO DE UN SISTEMA DE INFORMACIÓN DE SOPORTE A LA GESTIÓN DE LECTURA DE MEDIDORES BASADA EN SERVICIOS WEB PARA LA OTB SIRPITA”**

Presentado para optar el Grado Académico de:

LICENCIADO EN INGENIERÍA DE SISTEMAS

Autor: Mendoza Fuentes María Eugenia

Tutor: Ing. Luis Marcelo Garay Choqueribe

Cochabamba, Septiembre de 2016

Dedicatoria

Agradecimientos

Reconocimientos

Recomendaciones (Opcional)

Resumen Ejecutivo

Índice de Contenido

Índice de Tablas

Indice de Graficos

Índice de Ilustraciones

Indice de apendices

Indice anexos

Glosario de terminos

Contenido

[1 Introducción 3](#_Toc461130364)

[2 Introduccion Antecedentes 1](#_Toc461130365)

[3 Objeto de Estudio 1](#_Toc461130366)

[4 Beneficiarios del proyecto 1](#_Toc461130367)

[5 Problema 1](#_Toc461130368)

[5.1 Identificación del Problema 1](#_Toc461130369)

[5.2 Formulación del Problema 1](#_Toc461130370)

# Introducción

Las Organizaciones Territoriales de Base se consideran como tales a las juntas vecinales que constituyen organizaciones de derecho privado, para su constitución y funcionamiento gozan de independencia respecto a las entidades u órganos públicos, en el marco de las normas legales que rigen la materia.

La OTB Sirpita es una junta de vecinos ubicada en la zona Tiquipaya, de la ciudad de Cochabamba, cuya labor es agrupar a vecinos del entorno cuyo interés está vinculado con la asociación AGUA POTABLE, responsable del registro y administración del servicio de agua potable en la zona.

La propuesta está circunscrita a mejorar la forma de administración del consumo de agua potable dentro la OTB; de tal manera, que exista una coordinación para el cumplimiento de nuestros objetivos y así dando una solución a la problemática de la toma de lecturas para todos los socios.

Se implementará servicios web al modelo de trabajo dentro de la OTB ya que el uso principal es permitir la comunicación entre la empresa y sus clientes con el internet, siendo así que los web services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos sistemas de información.

Los Web Services han crecido en popularidad y han comenzado a mejorar los procesos de negocios, y están construidos con varias tecnologías que trabajan conjuntamente con los estándares que están emergiendo para asegurar la seguridad y operatividad, de modo de hacer realidad que el uso combinado de varios Web Services, independiente de la o las empresas que los proveen, este garantizado.

El desarrollo del proyecto se apropia a la arquitectura de servicios web dando una interoperabilidad e independencia en el intercambio de datos siendo así la implementación de servicios web al modelo de trabajo dentro de la OTB ya que el uso principal es permitir la comunicación entre la empresa y sus clientes con el internet, siendo así que los web services permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivos sistemas de información.

# CAPITULO I. Marco Referencial

# Antecedentes

La OTB Sirpita desde el año de su fundación 13 Mayo del 1954 se ha constituido como junta, desde el cual presenta el manejo de datos en relación del cobro y lectura de agua potable, siendo así que desde muchos años los datos están registrados en documentos impresos que pueden ser propensos a perdidas como también dificulta la información en los reportes de gastos o ingresos de efectivo de dinero generado por el cobro de agua.

# Objeto de Estudio

Gestionar el modelo de administración del consumo de agua dentro la OTB, Estudio de creación de una página web para ser utilizado como información de la Junta Vecinal y creación de un Web Services.

# Beneficiarios del proyecto

Los beneficiarios directos son el equipo de desarrollo, administradores y nuestros beneficiarios indirectos llegan a ser los clientes y vecinos.

# Problema

## Identificación del Problema

Actualmente en la OTB su sistema de administración no cumple con los requerimientos y se puede evidenciar el trabajo del cobro, información de socio sobre su consumo de agua, meses adeudados y toda información se encuentra administrada solo con documentación física, siendo así que para los encargados que tienen que utilizar ó actualizar cierta información de estas bases de datos, la tarea es muy tediosa pues se tiene que buscar un cierto archivo de forma manual y tener toda la información solo en documentos puede ser motivos de pérdidas.

## Formulación del Problema

¿Cómo se puede administrar, visualizar, optimizar la gestión de información dentro de la OTB Sirpita e implementar tecnologías para usuarios finales?

# Objetivos del Proyecto

## Objetivo General

Desarrollar un sistema de información basada en Servicios Web para el apoyo a la gestión de Lectura de Medidores en la OTB Sirpita

## Objetivos Específicos

1. Obtener Requerimientos Funcionales y No Funcionales
2. Diseñar e implementar Base de Datos
3. Implementar los Servicios Web
4. Gestionar el procedimiento de la Información del consumo de agua.
5. Gestionar el procedimiento de toma de lecturas.
6. Realizar Módulo de cobro del agua y registros de socios
7. Realizar módulo de registro de Socios
8. Realizar módulo de pagos de consumo de Agua.

# Justificación del Proyecto

## Justificación Técnica

Implementar Web Services compartiendo la lógica del negocio, los datos y los procesos, por medio de una interfaz a través de la red; agregando así a la Web Services la interfaz para usuarios, mediante una página Web que ayude al tema de administración de la toma de lecturas de agua, pagos, etc., tal de entregarle a los usuarios un funcionalidad específica que provee un determinado Web Service.

## Justificación social

La OTB tendrá un avance tecnológico a través de la creación de la página web para visualizar la información del consumo de agua e información del socio accediendo a través del servicio web beneficiando a la comunidad de Sirpita.

# Aportes

## Aporte a la Ingeniería

Implementación de comunicación de un web service y el sistema web en el módulo de consumo del agua potable. La facilidad de actualización de información con otros dispositivos será de forma inmediata y no por medio de terceros en el área de software en el área de software.

## Aporte a la empresa

Proporcionar el sistema al área de administración para la toma de lectura en los medidores de la OTB.

# Alcance del proyecto

El presente proyecto contempla todas las instalaciones necesarias para la ejecución de las distintas fases del proceso de desarrollo del servicio web del módulo para el registro de toma de lectura de agua.

## Temático

El estudio estará orientado a la administración de control de consumo de agua de la OTB.

## Geográfico

Este estudio tendrá como espacio territorial la comunidad de Sirpita del municipio de Tiquipaya de la ciudad de Cochabamba.

## Social

El proyecto beneficia a todos los socios registrados en la asociación de agua potable con el acceso al sistema 24/7 a través de internet.

# Factibilidad

## Factibilidad Técnica

Se dispone de los conocimientos y habilidades en el manejo de métodos, procedimientos y funciones requeridas para el desarrollo e implantación del proyecto. Como también se dispone del equipo y herramientas para llevarlo a cabo. Quien es mi equipo??

## Factibilidad Financiera

Se dispone del capital en efectivo o de los créditos de financiamiento necesario para invertir en el desarrollo del proyecto.

## Factibilidad Operativa

Existe el personal para el manejo del sistema, será capacitado para llevar a cabo el proyecto y así mismo, como también usuarios finales dispuestos a emplear los servicios generados por el sistema desarrollado.

## Factibilidad de Fechas

Tomará un tiempo de 6 meses.

# CAPITULO II. Marco teórico conceptual

## Qué es una Metodología ágil

Lo ágil se define como la habilidad de responder de forma versátil al cambio para maximizar los beneficios. Las metodologías ágiles varían en su forma de responder al cambio, pero en general comparten las siguientes características:

* Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
* El software que funciona es más importante que la documentación exhaustiva.
* La colaboración con el cliente en lugar de la negociación de contratos.
* La respuesta al cambio en lugar de aferrarse a un plan.

## Metodologías ágiles vs. Tradicionales

El siguiente cuadro recoge esquemáticamente estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada una de estas filosofías de desarrollo de software.

|  |  |
| --- | --- |
| **Metodologías Ágiles** | **Metodologías Tradicionales** |
| La planificación del trabajo sólo comprende el ciclo en el que se está trabajando (normalmente 30 días). | Trabajo y gestión guiada por un plan general del proyecto que comprende todo su ciclo de desarrollo. |
| Descubrimiento progresivo de requisitos, e incorporación de cambios en cualquier iteración del desarrollo. | Conocimiento detallado de los requisitos antes de comenzar el diseño del proyecto. |
| “Refactorización” de código como modelo de trabajo compatible con el punto anterior. | “Hacerlo bien a la primera”. Evitar la re-codificación y el re-trabajo que supone una pérdida de eficiencia. |
| Comunicación directa entre los integrantes del equipo (incluidos cliente y usuarios) prefiriendo la verbal directa. | Comunicación formal según el plan de comunicación del proyecto. |
| Equipos auto-gestionados. | Gestión de equipos y personas centralizada en el gestor del proyecto. |
| No existe contrato tradicional o al menos es bastante flexible. | Existe un contrato prefijado. |
| El cliente es parte del equipo de desarrollo. | El cliente interactúa con el equipo de desarrollo mediante reuniones. |
| Grupos pequeños (hasta 20 integrantes) y trabajando en el mismo sitio. | Grupos grandes y posiblemente distribuidos. |
| Pocos artefactos. | Más artefactos. |
| Pocos roles. | Más roles. |
| Menos énfasis en la arquitectura del software. | La arquitectura del software es esencial y se expresa mediante modelos |

Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus adeptos y quienes por prejuicio no las ven como alternativa a las metodologías tradicionales.

## Scrum

Scrum es una metodología ágil de gestión de proyectos cuyo objetivo primordial es elevar al máximo la productividad de un equipo. Como método, Scrum enfatiza valores y prácticas de gestión, sin pronunciarse sobre requerimientos, prácticas de desarrollo, implementación y demás cuestiones técnicas. Más bien delega completamente en el equipo la responsabilidad de decidir la mejor manera de trabajar para ser lo más productivos posibles.

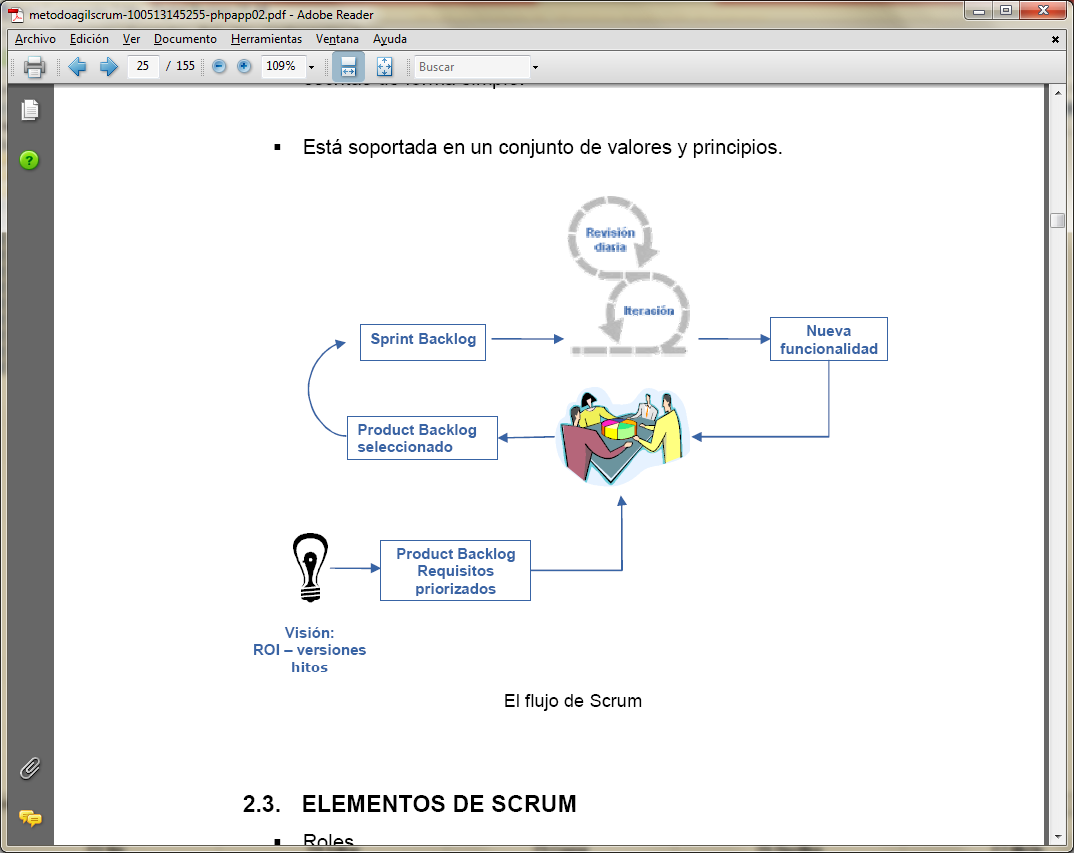
La palabra Scrum procede de la terminología del juego de rugby, donde designa al acto de preparar el avance del equipo en unidad pasando la pelota a uno y otro jugador. Igual que el juego, Scrum es adaptable, ágil, auto-organizante y con pocos tiempos muertos.

Se basa en los principios ágiles:

* Privilegiar el valor de la gente sobre el valor de los procesos.
* Entregar software funcional lo más pronto posible.
* Predisposición y respuesta al cambio.
* Fortalecer la comunicación y la colaboración.
* Comunicación verbal directa entre los implicados en el proyecto.
* Simplicidad; supresión de artefactos innecesarios en la gestión del proyecto.

**LA ESENCIA DE SCRUM**

* Más que una metodología de desarrollo es para gestionar proyectos, no contiene definiciones en áreas de ingeniería.
* Con visión de que el trabajo es efectuado por equipos auto-organizados y auto-dirigidos, logrando motivación, responsabilidad y compromiso.
* Está basada en un proceso constructivo iterativo e incremental donde las iteraciones tienen duración fija.
* Contiene definición de roles, prácticas y productos de trabajo escritas de forma simple.
* Está soportada en un conjunto de valores y principios.



**ELEMENTOS DE SCRUM**

* **Roles**
* Product Owner
* Scrum Master
* Team (Equipo)
* **Poda de requerimientos**
* **Product Backlog**
* **Sprint**
* Planificación
* Sprint Backlog
* Scrum
* Estimaciones
* Builds continuos
* Revisión del Sprint
* Reunión retrospectiva
* **Valores**
* Foco, comunicación, respeto y coraje.

**ROLES**

Scrum tiene una estructura muy simple. Todas las responsabilidades del proyecto se reparten en 3 roles:

* **Product owner (Dueño del producto)**

Representa a todos los interesados en el producto final. Sus áreas de responsabilidad son:

* Financiación del proyecto
* Requisitos del sistema
* Retorno de la inversión del proyecto
* Lanzamiento del proyecto

Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto (Product Backlog).

Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.

* **Scrum Master (Líder del proyecto)**

Responsable del proceso Scrum, de cumplir la meta y resolver los problemas. Así como también, de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que progrese según lo previsto.

Interactúa con el cliente y el equipo. Coordina los encuentros diarios, y se encarga de eliminar eventuales obstáculos. Debe ser miembro del equipo y trabajar a la par.

* **Team (Equipo)**

Responsable de transformar el Backlog de la iteración en un incremento de la funcionalidad del software. Tiene autoridad para reorganizarse y definir las acciones necesarias o sugerir remoción de impedimentos.

* Auto-gestionado
* Auto-organizado
* Multi-funcional

La dimensión del equipo total de Scrum no debería ser superior a veinte.

El número ideal es diez, más o menos dos. Si hay más, lo más recomendable es formar varios equipos, No hay una técnica oficial para coordinar equipos múltiples.

Scrum diferencia claramente entre estos dos grupos para garantizar que quienes tienen la responsabilidad tienen también la autoridad necesaria para poder lograr el éxito, y que quienes no tienen la responsabilidad, los observadores externos, no produzcan interferencias innecesarias.

|  |  |
| --- | --- |
| **Comprometido en el proyecto** | **Implicados en el proyecto** |
| * Dueño del producto * Equipo * Scrum Master | * Marketing * Comercial * Etc. |

**PODA DE REQUERIMIENTOS**

La primera actividad es armar una lista exhaustiva de los requerimientos originales del sistema. Luego se procede a ver qué requerimientos son realmente necesarios, cuáles pueden posponerse y cuáles eliminarse. Para ello debe identificarse un representante con capacidad de decisión, priorizar los requerimientos en base a su importancia y acordar cuáles son los prioritarios para la fecha de entrega.

La poda de requerimientos es una buena práctica implícita en modelos ágiles, se hace lo que el cliente realmente desea, no más.

**PRODUCT BACKLOG**

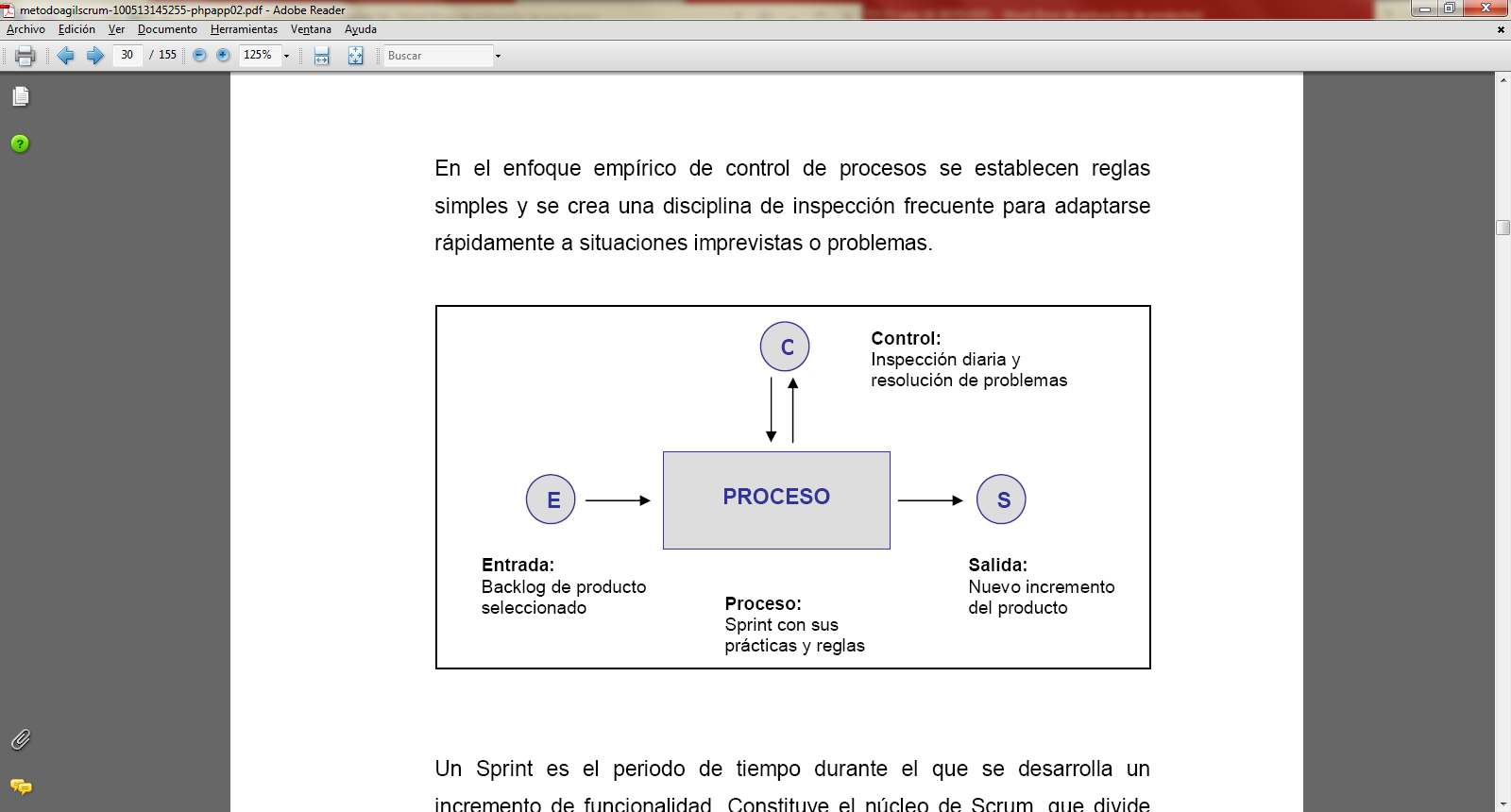
Con los requerimientos priorizados y podados armamos el Backlog de Producto. Este es una forma de registrar y organizar el trabajo pendiente para el producto (actividades y requerimientos).

Es un documento dinámico que incorpora constantemente las necesidades del sistema. Por lo tanto, nunca llega a ser una lista completa y definitiva. Se mantiene durante todo el ciclo de vida (hasta la retirada del sistema) y es responsabilidad del Product Owner.

**SPRINT**

Scrum está basado en el control empírico de procesos. Se utiliza cuando la capacidad de predicción es vaga, la incertidumbre alta o el proceso es demasiado complejo para ser modelado y definido.

En el enfoque empírico de control de procesos se establecen reglas simples y se crea una disciplina de inspección frecuente para adaptarse rápidamente a situaciones imprevistas o problemas.



Un Sprint es el periodo de tiempo durante el que se desarrolla un incremento de funcionalidad. Constituye el núcleo de Scrum, que divide de esta forma el desarrollo de un proyecto en un conjunto de pequeñas “carreras”.

**Reunión diaria**

**Backlog asignado al equipo**

**30 días**

**Sprint Backlog**

**Product Backlog priorizado por el propietario del producto**

**Nueva Funcionalidad demostrable**

* Duración máxima del Sprint: 30 días.
* Durante el Sprint no se puede modificar el trabajo que se ha acordado en el Backlog.
* Sólo es posible cambiar el curso de un Sprint, abortándolo, y sólo lo puede hacer el Scrum Master si decide que no es viable por alguna de las razones siguientes:

- La tecnología acordada no funciona.

- Las circunstancias del negocio han cambiado.

- El equipo ha tenido interferencias.

* **Planificación**

Se planifica en detalle el trabajo al inicio de cada Sprint asumiendo que los objetivos no van a cambiar durante el mismo. De esta manera se atenúa el riesgo.

Aspectos a tener en cuenta sobre la planificación de un Sprint:

· Una determinación general de alcance, frecuentemente basada en una EDT (Estructura de División del Trabajo).

· Estimaciones de esfuerzo de alto nivel realizadas durante la etapa de concepción del proyecto.

· Esfuerzo dedicado a labores de soporte o de preparación de los ambientes requeridos por el proyecto.

· Esfuerzo asociados a las reuniones diarias, de planificación y de revisión.

· Requerimientos de recursos de infraestructura o logísticos (máquinas, redes, licencias, papel, pizarras, etc.).

· Habilidades presentes y necesarias en el equipo.

· Restricciones asociadas al conocimiento del negocio, la tecnología o externas (legales, reglamentarias, estándares, etc.).

**Rol del Scrum Master durante la planificación:**

* Dirige la planificación.
* Es vínculo entre el equipo y el Product Owner del proyecto.
* Registra problemas y riesgos detectados durante la planificación.
* Registra las tareas, asignaciones y estimaciones.
* Inicia el Backlog del Sprint.
* **Sprint Backlog**

Trabajo o tareas determinadas por el equipo para realizar en un Sprint.

* Tareas a convertir en producto funcional.
* Las tareas se estiman en una duración entre 1 a 20 horas de trabajo. Las de mayor duración deben intentar descomponerse en sub-tareas de ese rango de tiempo.
* La estimación se actualiza día a día.
* **Scrum diario**

Scrum asume que el proceso es complejo y que es necesario inspeccionarlo frecuentemente, por eso se realiza una reunión diaria de seguimiento. El encuentro diario impide caer en el dilema señalado por Fred Brooks: “¿Cómo es que un proyecto puede atrasarse un año? Un día a la vez”.

El foco de la reunión es determinar el avance en las tareas y detectar problemas o “bloqueos” que estén haciendo lento el progreso del equipo o que eventualmente impidan a un equipo cumplir con la meta del Sprint. La idea es que ningún problema quede sin resolver o, por lo menos, sin iniciar alguna acción de respuesta dentro de las 24 horas después de su detección.

La reunión es además un espacio definido para que cada miembro del equipo comunique a los demás el estado de su trabajo y por lo tanto reafirme el compromiso.

**Rol del Scrum Master durante el Scrum:**

* Dirige la reunión y mantiene el foco.
* Hace preguntas para aclarar dudas.
* Registra (escribe o documenta) los problemas para su resolución después de la reunión.
* Se asegura que los miembros cuenten con el ambiente adecuado para la reunión.

## Testing

El Comité internacional de cualificación de pruebas (ISTQB: Internacional Software Testing Qualification Board, www.istqb.org) es una organización sin ánimo de lucro, creada en el año 2002 por empresas, instituciones, organizaciones y personas especializadas en el campo de las pruebas y la industria del software. El fin de la asociación es dar soporte y definir un esquema de certificación internacional. Dicho comité suministra el plan de estudios y el glosario en los cuales se definen los estándares internacionales por niveles y se establecen las guías para la acreditación y evaluación de los profesionales del testing a cargo de los comités de cada país. En España el SSTQB (o Comité Español de Testing) es el representante oficial del ISTQB.

Un servidor de integración continua se encarga de ejecutar una serie de procedimientos automatizados que chequean el código en busca de errores, aplicando métricas (sintaxis del código, etc.) y realizando tareas regulares como la documentación.

**Tipos de pruebas:**

• **Pruebas unitarias:** se encargan de probar una clase en concreto, testeando cada uno de sus métodos y viendo si dados unos parámetros de entrada, la salida es la esperada.

• **Pruebas funcionales:** como su propio nombre indican, prueban una funcionalidad completa, donde pueden estar implicadas una o varias clases, la propia interfaz de usuario y, en el caso del desarrollo web, llamada AJAX.

• **Pruebas de regresión:** son aquellas pruebas cuyo objetivo es comprobar por qué ha dejado de funcionar algo que ya funcionaba. El objetivo de las pruebas de regresión es no tener que “volver atrás”.

• **Pruebas de aceptación:** son pruebas funcionales, pero vistas directamente desde el cliente. Digamos que son aquellas pruebas que demuestran al cliente que la funcionalidad está terminada y funciona correctamente.

• **Pruebas de integración:** conjunto de pruebas unitarias, funcionales, de regresión y/o de aceptación que se realizan las probar el software. Incluye también comprobar que lo programado por los diferentes desarrollados no “choca” entre sí y que funcionará en un entorno real.

## Bootstrap

Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web.

Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT en el año 2011 y su desarrollo continua en un repositorio de GitHub.

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

Desde la aparición de Bootstrap 3 el framework se ha vuelto bastante más compatible con desarrollo web responsive, entre otras características se han reforzado las siguientes:

Soporte bastante bueno (casi completo) con HTML5 y CSS3, permitiendo ser usado de forma muy flexible para desarrollo web con unos excelentes resultados.

Se ha añadido un sistema GRID que permite diseñar usando un GRID de 12 columnas donde se debe plasmar el contenido, con esto podemos desarrollar responsive de forma mucho más fácil e intuitiva.

## Responsive

Debido a la proliferación de smartphones y tablets en el mercado actual, existe más diversidad que nunca de formatos de pantalla. De acuerdo con el estudio realizado por Comscore, las ventas de smartphone superarán a las de computadoras de escritorio durante este año y la adopción de tablets en Estados Unidos se prevee que experimente un crecimiento del 40% en los próximos 4 años, alcanzando los 75.8 millones en 2016.

Este panorama obliga a adaptar los formatos web a estos nuevos dispositivos y la estructura de cada uno de ellos, es indiscutible que necesitamos websites inteligentes que se adapten a todos ellos. Sobre todo si tenemos en cuenta las ventas a través de m-commerce (e-commerce adaptados a móvil), según eMarketer, las ventas a través de smartphones han crecido en un 81%.

A partir de todo esto, el término “responsive web design” se escucha frecuentemente, pero ¿qué es exactamente? El responsive design corresponde a una tendencia de creación de páginas web que pueden ser visualizadas perfectamente en todo tipo de dispositivos, desde ordenadores de escritorio hasta smartphones o tablets. Con este tipo de diseño no necesitas tener una versión para cada dispositivo, una sola web se adapta a todos ellos.

¿Cómo funciona el responsive web design?

En lugar de construir una website para cada tipo de dispositivo (smartphone, tablet, ordenador desktop, laptop e incluso, hoy en día, para smart TV), se crea una sola website utilizando CSS3 media queries y un layout con imágenes flexibles. De esta forma, la website detecta desde qué clase de dispositivo está accediendo el usuario y muestra la versión más optimizada para ese medio, reorganizando los elementos de la web e incluso discriminando algunos de ellos (menos imágenes, más ligeras, redistribución de las columnas en el diseño, menos texto, etc.).

El HTML5 permite una experiencia excelente para los usuarios, sin el coste de desarrollar una app nativa para cada dispositivo. En resumen, se diría que el responsive design son todo ventajas.

¿Cómo funcionan los editores web para responsive design?

A partir de plantillas personalizables creadas en HTML5 y CSS3 con fluid grids que re-colocan los elementos de la web, dependiendo del espacio disponible según la resolución de pantalla. De esta forma discriminan elementos, texto, pestañas del menú, reducen las imágenes de manera que la navegación sea fluida, intuitiva y se enriquezca la experiencia de usuario.

## Git Desktop

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar tus repositorios remotos. Los repositorios remotos son versiones de tu proyecto que se encuentran alojados en Internet o en algún punto de la red. Puedes tener varios, cada uno de los cuales puede ser de sólo lectura, o de lectura/escritura, según los permisos que tengas. Colaborar con otros implica gestionar estos repositorios remotos, y mandar (push) y recibir (pull) datos de ellos cuando necesites compartir cosas.

## Git Extension

Extensiones de Git es un conjunto de herramientas encaminadas a hacer trabajar con Git en Windows más intuitivo (tenga en cuenta que las extensiones de Git también está disponible en Linux y Macintosh OS X utilizando Mono). La extensión de shell se integrará en el Explorador de Windows y presenta un menú contextual de archivos y directorios. También hay un estudio de plug-in Visual usar Git desde el IDE de Visual Studio.

## Smart Git

SmartGit es una herramienta multiplataforma de entorno fácil e intuitivo que nos ayuda con Git, es un sofware de origen alemán desarrollado en Java que facilita el acceso a los repositorios GIT/SVN en forma visual, sin líneas de comandos. Tiene una versión Community gratuita y funciona en varios entornos operativos (Windows, MacOS, Linux/Unix).

## Github, Bitbucket y Stash

**Comparativa entre Github y Bitbucket**

Github y Bitbucket son dos de los mayores servicios de hosting basados en la web para los proyectos de código fuente y de desarrollo. Los dos no son compatibles con el sistema de control de versiones SVN populares (VCS). Y ellos tienen diferentes enfoques de repositorios públicos vs. privado.

## Github

En primer lugar, Github solamente acoge proyectos que utilizan el Git VCS. Eso es. Nada más. Pero Git es de lejos el VCS más utilizados, por lo Github sigue siendo el más grande de acogida código de todos ellos, con unos 13,7 millones de repositorios-plus de código.

Github es gratuito para los repositorios públicas ya diferencia Bitbucket, no ofrece repositorios privado gratuito. Para repositorios privadas, Github permite un número ilimitado de colaboradores agrupados en los siguientes planes de pago. Los planes de la Organización permiten que un administrador central que maneja equipos y pueden establecer diferentes niveles de permisos.

## Bitbucket

La ventaja de Bitbucket sobre Github es que soporta el Mercurial VCS además de Git. Pero también no es compatible con SVN, todavía. Bitbucket está escrito en Python y usa el framework web Django.

Bitbucket integra muy bien con JIRA, un proyecto popular y aplicación tema-gestión.

## Servidor Web (Web Services)

Servicios web (web services) son un nuevo modelo de desarrollo de aplicaciones distribuidas basadas en Internet

• Un servicio web es un componente al que se puede acceder mediante protocolos Web estándar:

– Los mensajes para invocar el servicio se codifican en XML

– Los mensajes se pueden transportar utilizando HTTP o cualquier otro protocolo de transporte estándar

• Estándar de W3C

– XML

– SOAP

Características:

– Permiten que las aplicaciones compartan información de forma remota

– Permiten integrar sistemas heterogéneos de manera barata y fiable

– Son unidades de código discretas, cada una de las cuales se encarga de un conjunto limitado de tareas.

– Están basados en XML, el lenguaje estándar de intercambio de información en Internet

– Pueden utilizarse en cualquier plataforma o sistema operativo, independientemente del

## Lenguaje Php

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Las páginas de PHP contienen HTML con código incrustado que hace "algo". El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP".

Lo que distingue a PHP de algo del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas.

MySQL es muy utilizado en aplicaciones web, Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones Web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante adelantar monitoreos sobre el desempeño para detectar y corregir errores tanto de SQL como de programación.

## Sistemas de Información

Un sistema de Información es un conjunto de elementos Orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior. Generados para cubrir una necesidad o un objetivo.

# RUP (Rational Unified Process o Proceso Unificado Racional)

Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos; un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

## CARACTERÍSTICAS

RUP cuenta con tres características principales:

***Dirigido por los Casos de Uso***

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo.

Los Casos de Uso no sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer una relación entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

***Centrado en la arquitectura***

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. Involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo.

La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema. RUP presta atención a atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Cada modelo tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Es conveniente ver el sistema desde diferentes perspectivas para comprender mejor el diseño por lo que la arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, abstrayéndose de los demás. Para RUP, todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, el cual recibe este nombre porque lo forman las vistas lógica, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas.

Al final de la fase de elaboración se obtiene una primera aproximación de la arquitectura donde fueron seleccionados una serie de Casos de Uso arquitectónicamente relevantes.

***Iterativo e incremental***

Debe existir un equilibrio entre los Casos de Uso y la arquitectura que es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo.

Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. Una iteración puede realizarse por medio de una cascada de etapas como se muestra en la Figura 6. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

## FASES E ITERACIONES DE RUP

El ***ciclo de vida RUP*** es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades.

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una primera aproximación o línea base de la arquitectura.

***Fase de inicio:***las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

***Fase de elaboración:*** las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la línea base de la arquitectura.

***Fase de construcción:***se lleva a cabo la construcción del producto por medio de una serie de iteraciones (implementación, pruebas y muestra del sistema). Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación deseada de la nueva versión del producto.

***Fase de transición:***se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios con el fin de que la prueben.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina o actividad varía.

En la siguiente figura se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

En RUP se identifican 6 prácticas con las que define una forma efectiva de trabajar para los equipos de desarrollo de software.

**Gestión de requisitos:** RUP brinda una guía para encontrar, organizar, documentar, y seguir los cambios de los requisitos funcionales y restricciones. Utiliza una notación de Caso de Uso y escenarios para representar los requisitos.

**Desarrollo de software iterativo:** Desarrollo del producto mediante iteraciones con hitos bien definidos, en las cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto.

**Desarrollo basado en componentes:** La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o se desarrollan sus componentes.

**Modelado visual (usando UML):** UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema software. Es un estándar de la OMG. Utilizar herramientas de modelado visual facilita la gestión de dichos modelos, permitiendo ocultar o exponer detalles cuando sea necesario. El modelado visual también ayuda a mantener la consistencia entre los artefactos del sistema: requisitos, diseños e implementaciones. En resumen, el modelado visual ayuda a mejorar la capacidad del equipo para gestionar la complejidad del software.

**Verificación continúa de la calidad:** Es importante que la calidad de todos los artefactos se evalúe en varios puntos durante el proceso de desarrollo, especialmente al final de cada iteración. En esta verificación las pruebas juegan un papel fundamental y se integran a lo largo de todo el proceso. Para todos los artefactos no ejecutables las revisiones e inspecciones también deben ser continuas.

**Gestión de los cambios:** Los cambios son un factor de riesgo crítico en los proyectos de software. Los artefactos de software cambian no sólo debido a acciones de mantenimiento posteriores a la entrega del producto, sino que durante el proceso de desarrollo, especialmente importantes por su posible impacto son los cambios en los requisitos.

# UML (Unified Modeling Language o Lenguaje Unificado de Modelado)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

UML cuenta con 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil ordenarlos de la siguiente manera:

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

Diagrama de clases: Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. Atributos también llamados propiedades o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. **Operaciones** comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. **Interfaz** es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo. **Herencia** se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona, un transporte, un documento o un paquete. Estos ejemplos de clases de objetos reales, es sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales como datos que corresponden al objeto.

Diagrama de componentes:Representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Debido a que estos son más parecidos a los diagramas de casos de usos estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

Diagrama de objetos:Se puede considerar un caso especial de un diagrama de clases en el que se muestran instancias específicas de clases en un momento particular del sistema. Los diagramas de objetos utilizan un subconjunto de los elementos de un diagrama de clase. Los diagramas de objetos no muestran la multiplicidad ni los roles, aunque su notación es similar a los diagramas de clase. Una diferencia con los diagramas de clase es que el compartimiento de arriba va en la forma.

Diagrama de estructura compuesta:Es un tipo de diagrama de estructura estática en el Lenguaje de Modelado Unificado (UML), que muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posibles. Esto puede incluir partes internas, puertas mediante las cuales, las partes interactúan con cada una de las otras o mediante las cuales, instancias de la clase interactúan con las partes y con el mundo exterior, y conectores entre partes o puertas. Una estructura compuesta es un conjunto de elementos interconectados que colaboran en tiempo de ejecución para lograr algún propósito. Cada elemento tiene algún rol definido en la colaboración.

* **Parte:** Representa un rol jugado en tiempo de ejecución por una instancia de una clase o por una colección de instancias. La parte puede nombrar solamente un rol, una superclase abstracta, o puede nombrar una clase concreta específica. La parte puede incluir un factor de multiplicidad (cardinalidad), tal como el [0..\*] mostrado para Viewer en el diagrama.
* **Puerta:** Es un punto de interacción que puede ser usado para conectar clasificadores estructurados con sus partes y con el ambiente. Las puertas pueden opcionalmente especificar los servicios que proveen y los servicios que requieren de otras partes del sistema. En el diagrama, cada uno de los cuadrados pequeños es una puerta. Cada puerta tiene un tipo y esta etiquetado con un nombre, tal como "var", "indVar1", or "view" en el diagrama. Las puertas pueden contener un factor de multiplicidad. Las puertas pueden ya sea delegar los requerimientos recibidos a partes internas, o pueden entregarlos directamente para el comportamiento del clasificador estructurado en el que la puerta está contenido. Las puertas públicas que son visibles en el ambiente son mostradas sobre el borde (límite o frontera), mientras que las puertas protegidas que no son visibles en el ambiente son mostradas dentro de la frontera (borde o límite).
* **Conector:** Une dos o más entidades, permitiéndoles interactuar en tiempo de ejecución. Un conector es representado por una línea que une una combinación de partes, puertas y clasificadores estructurados. El diagrama muestra tres conectores entre puertas, y un conector entre un clasificador estructurado y una parte.

**Colaboración:** Es generalmente más abstracta que un clasificador estructurado. Ésta es mostrada como un óvalo sin relleno conteniendo los roles que las instancias pueden jugar en la colaboración.

**Clasificador estructurado:** Representa una clase, frecuentemente una clase abstracta, cuyo comportamiento puede ser completa o parcialmente descrito mediante interacciones entre partes. Un *Clasificador Encapsulado* es un tipo de clasificador estructurado que contiene puertas.

Diagrama de despliegue: Se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.

En UML los componentes ya no están dentro de nodos. En cambio, puede haber artefactos u otros nodos dentro de un nodo. Un artefacto puede ser algo como un archivo, un programa, una biblioteca, o una base de datos construida o modificada en un proyecto. Estos artefactos implementan colecciones de componentes. Los nodos internos indican ambientes, un concepto más amplio que el hardware propiamente dicho, ya que un ambiente puede incluir al lenguaje de programación, a un sistema operativo, un ordenador o un clúster de terminales.

La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema. Aunque UML no es un lenguaje de especificación hardware de propósito general, se ha diseñado para modelar muchos de los aspectos hardware de un sistema a un nivel suficiente para que un ingeniero software pueda especificar la plataforma sobre la que se ejecuta el software del sistema.

Diagrama de paquetes:Muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Con estas líneas maestras sobre la mesa, los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:

***Diagrama de actividades:***Representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un Diagrama de Actividades muestra el flujo de control general.

***Diagrama de casos de uso:***Define una notación gráfica para representar casos de uso. UML no define estándares para el formato de casos de uso, y así mucha gente no entiende que esta notación gráfica define la naturaleza de un caso de uso; sin embargo una notación gráfica puede solo dar una vista general simple de un caso de uso o un conjunto de casos de uso. Los diagramas de casos de uso son a menudo confundidos con los casos de uso. Mientras los dos conceptos están relacionados, los casos de uso son mucho más detallados que los diagramas de casos de uso. El valor verdadero de un caso de uso reposa en dos áreas: La descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio. Esta descripción se enfoca en el valor suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas. La posición o contexto del caso de uso entre otros casos de uso. Dado que es un mecanismo de organización, un conjunto de casos de uso coherente, consistente promueve una imagen fácil del comportamiento del sistema, un entendimiento común entre el cliente/propietario/usuario y el equipo de desarrollo.

Diagrama de estados: muestra la secuencia de estados por los que pasa bien un caso de uso, bien un objeto a lo largo de su vida, o bien todo el sistema. En él se indican qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera. En cuanto a la representación, un diagrama de estados es un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con los nombres de los eventos. Un estado se representa como una caja redondeada con el nombre del estado en su interior. Una transición se representa como una flecha desde el estado origen al estado destino.

Los ***Diagramas de Interacción*** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

Diagrama de secuencia:muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Mientras que el diagrama de casos de uso permite el modelado de una vista *business* del escenario, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

Diagrama de comunicación:Modela las interacciones entre objetos o partes en términos de mensajes en secuencia. Los diagramas de comunicación representan una combinación de información tomada desde el diagrama de clases, secuencia, y diagrama de casos de uso describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

Los diagramas de comunicación y de secuencia describen información similar, y con ciertas transformaciones, pueden ser transformados unos en otros sin dificultad. Para mantener el orden de los mensajes en un diagrama de comunicación, los mensajes son etiquetados con un número cronológico y colocado cerca del enlace por el cual se desplaza el mensaje.

Diagrama de tiempos:es una gráfica de formas de onda digitales que muestra la relación temporal entre varias señales, y cómo varía cada señal en relación a las demás. Son una representación especial de interacción que se enfoca en el tiempo de los mensajes enviados entre objetos. Se pueden usar estos diagramas para mostrar restricciones detalladas sobre el tiempo, ó para mostrar los cambios con líneas de vida respecto al tiempo. Los diagramas de tiempo son generalmente utilizados con sistemas en tiempo real o en sistemas embebidos.

Un cronograma puede contener cualquier número de señales relacionadas entre sí. Examinando un diagrama de tiempos, se puede determinar los estados, nivel alto o nivel bajo, de cada una de las señales en cualquier instante de tiempo especificado, y el instante exacto en que cualquiera de las señales cambia de estado con respecto a las restantes.

El propósito primario del diagrama de tiempos es mostrar los cambios en el estado o la condición de una línea de vida (representando una Instancia de un Clasificador o un Rol de un clasificador) a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o estímulos aceptados. Los eventos que se reciben se anotan, a medida que muestran cuándo se desea mostrar el evento que causa el cambio en la condición o en el estado.

Diagrama global de interacciones***:***Es un diagrama de comportamiento, más precisamente, uno de los cuatro diagramas de interacción. Muestra una cierta vista sobre los aspectos dinámicos de los sistemas modelados. Aunque un diagrama global de las interacciones es una representación gráfica de una interacción, éste se distingue fuertemente de los diagramas de secuencia y de comunicación, dos de los otros diagramas de interacción. De hecho, algunos elementos gráficos del diagrama global de las interacciones están tomados del diagrama de actividades, otro diagrama de comportamiento para el modelado de actividades.

Los modelos de interacción pueden llegar a ser muy grandes para sistemas complejos. Si el número de líneas de vida participantes y el número de mensajes intercambiados exceden una cierta medida, se impone “modularizar” las interacciones y dividir en partes pequeñas, más manejables, de acuerdo a principios universales del diseño de sistemas, que también pueden ser visualizadas con la ayuda de un clásico diagrama de secuencias. La visión de conjunto de toda la interacción, de manera que el cuadro global, puede entonces ser representada con la ayuda del diagrama global de las interacciones, provisto para eso.

# CAPITULO III. Ingeniería de Requerimientos

# METODOS DE RECOLECCION DE DATOS

ENTREVISTA

1.- Entrevista, a la Administradora

* ¿Requiere de un software que controle el consumo de agua potable?

Si, se requiere un sistema que ayude al control del consumo del agua, de fácil uso y que satisfaga las necesidades de la comunidad.

* ¿Cómo llega a controlar el consumo del agua dentro de la OTB?

El consumo se llega a controlar en documentos, todo es manual y que llegamos a controlar: registro de lecturas, consumos, pagos, etc. Documentos hechos por los responsables de la Comunidad.

* ¿Qué problemas existen actualmente por no tener los datos de los socios sobre el consumo del agua en un sistema?

Malos datos, no se llega a controlar todo, se pierde información, no se tiene información en el momento.

* ¿Qué tipo de requerimientos son los que desearía que se implemente en un software de control de consumo?

Que pueda contemplar:

Control de Socios

Registros de los Consumos

Pagos del consumo

Toma de lecturas

Reportes

Multas

Citaciones

* ¿Quiénes manejaran el sistema?

Se cuenta con una persona encargada para el manejo del sistema

* ¿Se requiere control de usuarios?

Si, que maneje permisos para los usuarios que manejen el software

* **HERRAMIENTAS**

DIAGRAMA DE CLASES

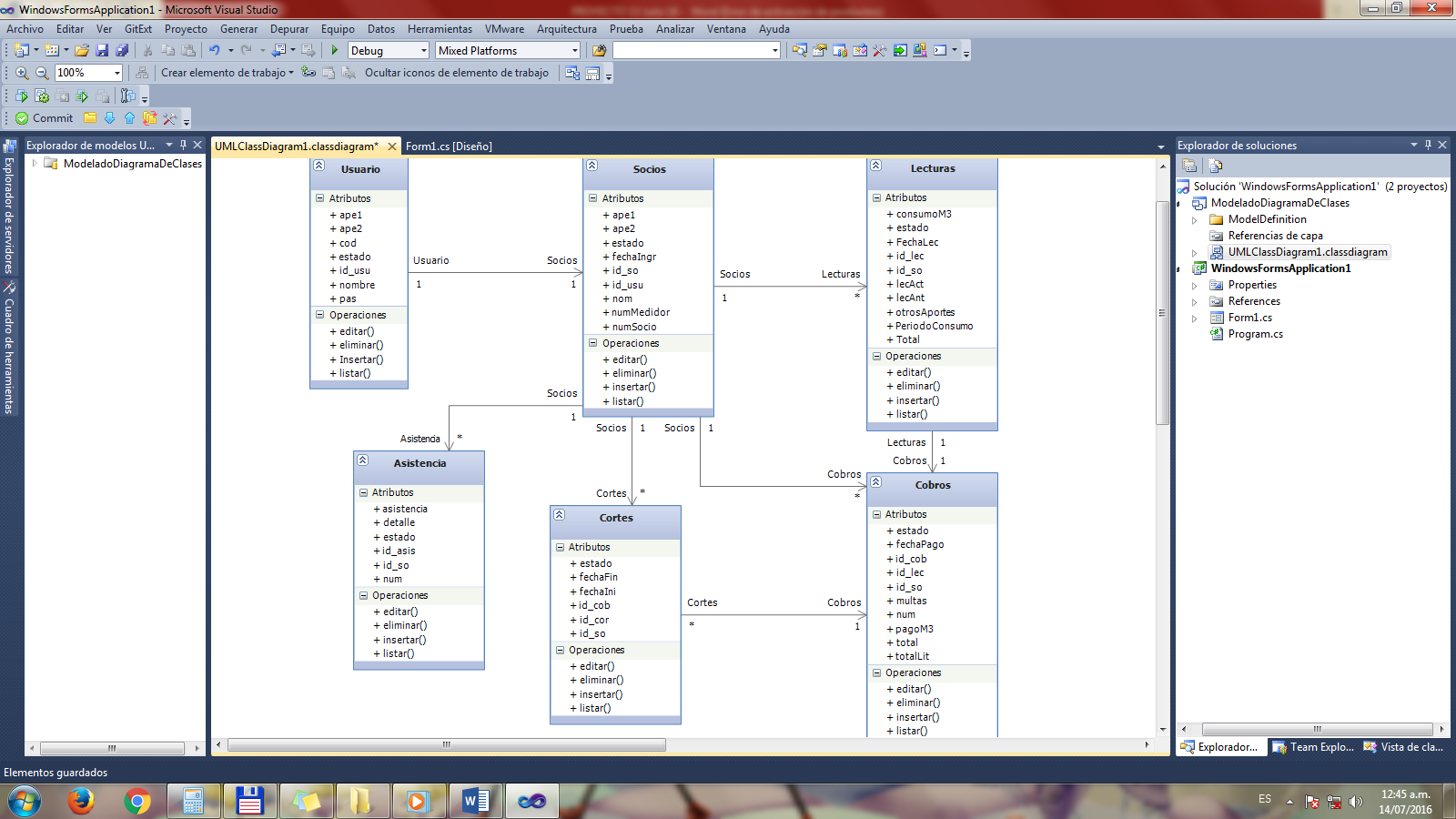


DIAGRAMA CASOS DE USO







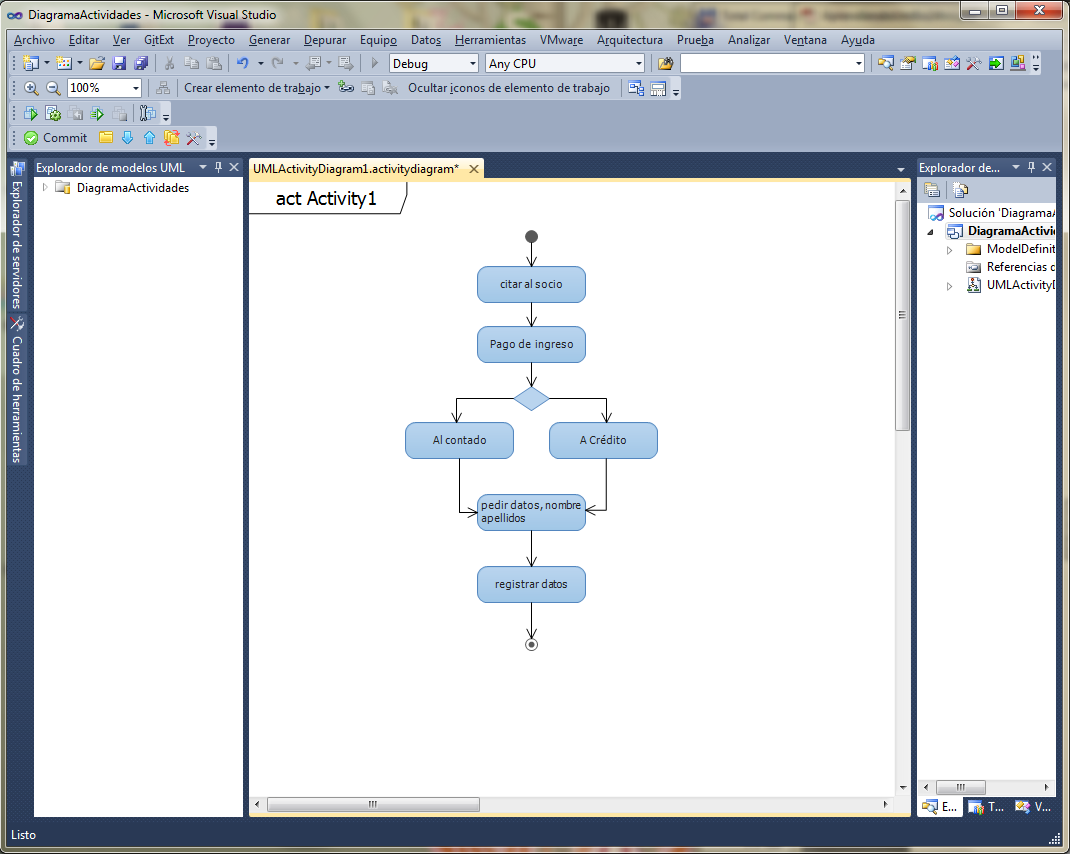






DIAGRAMA DE ACTIVIDADES

Proceso de registro de socios



Lecturas Aviso de cobranza

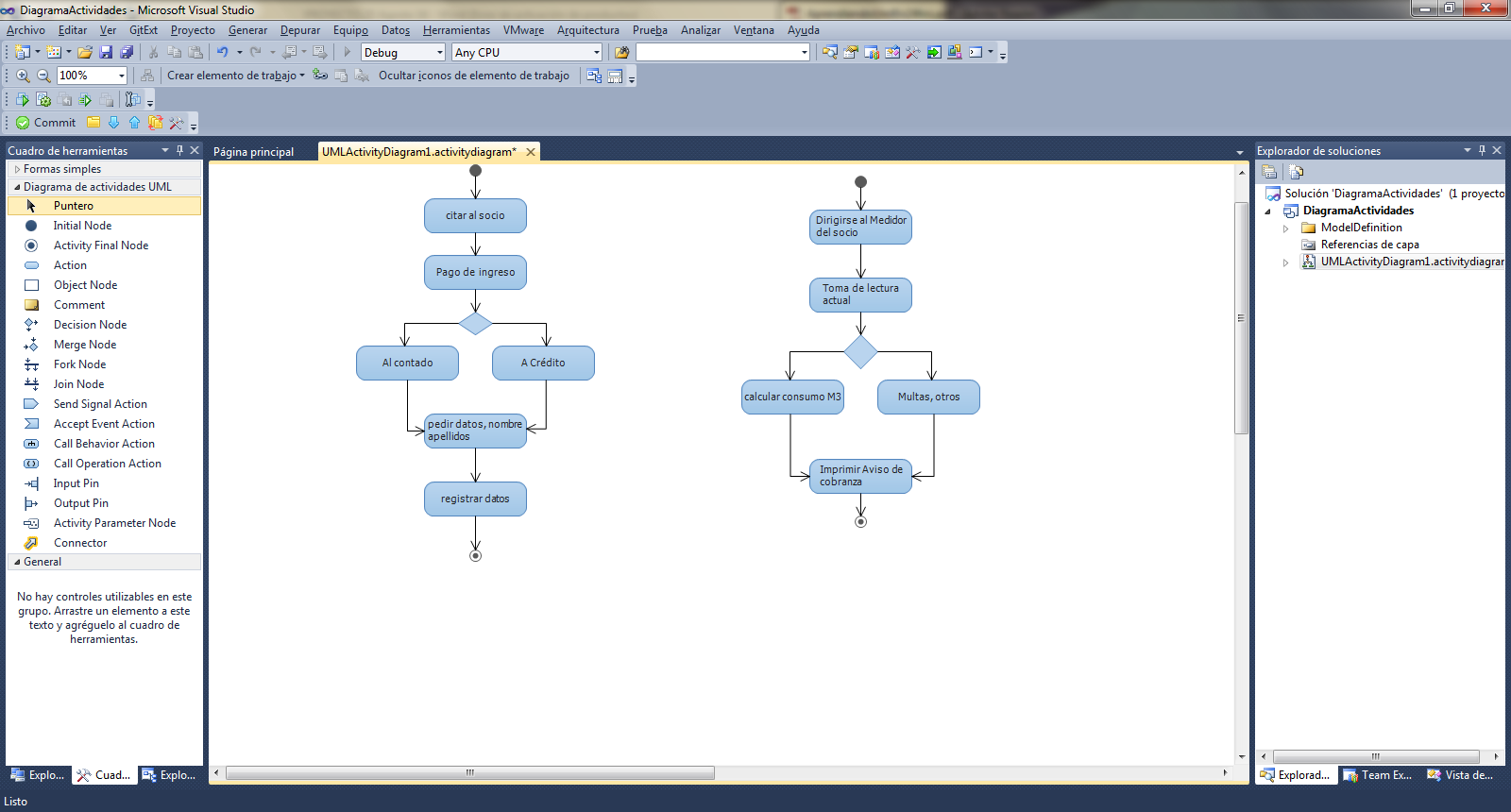
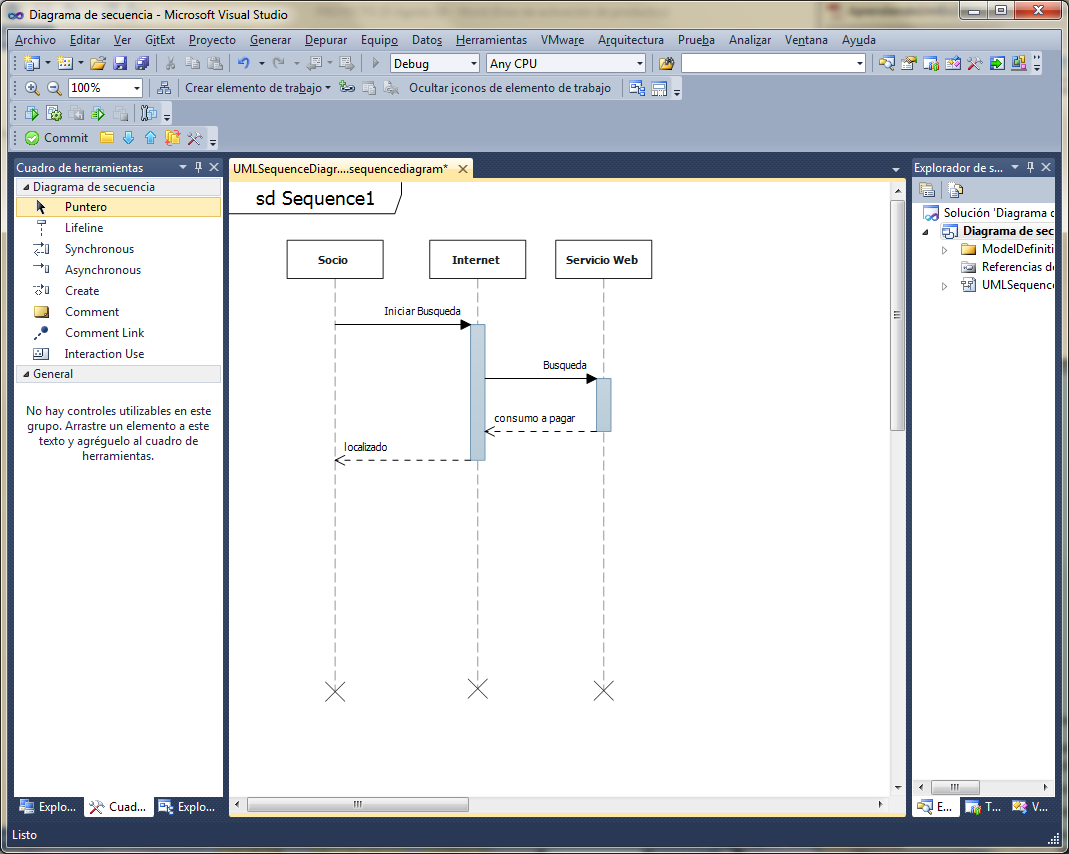


DIAGRAMA DE SECUENCIA

Consultar consumo de agua



# DISEÑO DE LA PROPUESTA

**Lista de requerimientos**

* control de Ingresos al sistema
* Administrar Socios
* Registro de toma de Lecturas
* Administrar cobranza
* Consultar pagos
* Consultar deudas
* Consultar Reuniones
* Administrar asistencias

## Características de los usuarios

**No comprendo que se hacer aquí?**

|  |  |
| --- | --- |
| **Tipo de usuario** | Usuario |
| **Formación** | Cualquier tipo de información |
| **Actividades** | Utilizar el sistema para comunicarse con otros usuarios |

|  |  |
| --- | --- |
| **Tipo de usuario** | Sistema |
| **Formación** |  |
| **Actividades** | Realizar la conexión habilitando puertos |

|  |  |
| --- | --- |
| **Tipo de usuario** | Servidor |
| **Formación** |  |
| **Actividades** | Responder a las solicitudes enviadas por el usuario |

.

# REQUISITOS ESPECÍFICOS

## REQUERIMIENTOS FUNCIONALES

# 

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF01 |
| **Nombre del Requerimiento:** | Ejecución de la página web |
| **Características:** | Los usuarios podrán acceder desde cualquier computadora |
| **Descripción del requerimiento:** | El podrá ser usado por cualquier usuario dependiendo del módulo en el cual se encuentre y su nivel de accesibilidad. |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF02 |
| **Nombre del Requerimiento:** | Registro de Usuarios |
| **Características:** | Los usuarios colocaran tendrán un usuario y contraseña para ingresar al sistema |
| **Descripción del requerimiento:** | Deberá permitir registrar usuarios, nombre y apellido, usuario, contraseña y tipo de usuario. |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

## 

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF03 |
| **Nombre del Requerimiento:** | Modificar Usuarios |
| **Características:** | Los usuarios podrán modificar, usuario y contraseña |
| **Descripción del requerimiento:** | Capturar los datos del usuario y poder modificar nombre y apellido, usuario, contraseña y el tipo de usuario |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF04 |
| **Nombre del Requerimiento:** | Eliminar Usuarios |
| **Características:** | Los usuarios podrán ser deshabilitados |
| **Descripción del requerimiento:** | Permitir eliminar usuarios, sin ninguna restricción para los administradores, excepto en el caso que el usuario haya realizado algún movimiento en el sistema |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF05 |
| **Nombre del Requerimiento:** | Registro de Socios |
| **Características:** | Socios registrados podrán acceder con sus cuentas al sistema |
| **Descripción del requerimiento:** | Permite registrar datos nombre, apellidos, teléfonos, Numero de Socio, Medidor, Pagos, consumos, fecha de ingreso, etc. |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF06 |
| **Nombre del Requerimiento:** | Modificar Socios |
| **Características:** | Permite editar datos del socios |
| **Descripción del requerimiento:** | Tener la posibilidad de modificar datos del socio, para poder actualizar datos o modificar cualquier equivocación. |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF07 |
| **Nombre del Requerimiento:** | Eliminar Socios |
| **Características:** | Poder eliminar socio |
| **Descripción del requerimiento:** | Permitir eliminar socios, en caso que no tenga registrados ningún movimiento con su nombre. |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF08 |
| **Nombre del Requerimiento:** | Registrar Consumo de Agua |
| **Características:** | Toma de lectura del medidor del socio |
| **Descripción del requerimiento:** | Debe permitir registrar lectura de consumo, registrar periodo de la lectura, capturar lectura anterior, multas y poder calcular total a cancelar por consumo |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF09 |
| **Nombre del Requerimiento:** | Modificar datos de consumo |
| **Características:** | Poder realizar cambios del consumo registrado |
| **Descripción del requerimiento:** | Realizar cambios a datos del consumo como ser, periodo de la lectura, multas y re calcular total a cancelar por consumo |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RF10 |
| **Nombre del Requerimiento:** | Eliminar datos de consumo |
| **Características:** | Permitir eliminar datos registrados |
| **Descripción del requerimiento:** | Debe permitir a datos del consumo como ser, periodo de la lectura, multas y re calcular total a cancelar por consumo |
| **Requerimiento NO funcional:** | * RNF01 * RNF02 * RNF05 |
| **Prioridad del requerimiento:** Alta | |

## REQUERIMIENTOS NO FUNCIONALES

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RNF01 |
| **Nombre del Requerimiento:** | Mantenimiento. |
| **Características:** | La página web deberá de tener un manual de usuario para facilitar los mantenimientos que serán realizados por el administrador. |
| **Descripción del requerimiento:** | El sistema debe disponer de una documentación fácilmente actualizable que permita realizar operaciones de mantenimiento con el menor esfuerzo posible. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RNF02 |
| **Nombre del Requerimiento:** | Diseño de la interfaz |
| **Características:** | La página web debe tener una interfaz de usuario, teniendo en cuenta las características de la institución. |
| **Descripción del requerimiento:** | La interfaz de usuario debe ajustarse a las características de la institución, dentro de la cual estará incorporado el sistema de gestión de procesos y el inventario. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RNF03 |
| **Nombre del Requerimiento:** | Desempeño |
| **Características:** | Garantizar a los usuarios un desempeño de la página web  Envió de datos confiabilidad. |
| **Descripción del requerimiento:** | Garantizar el desempeño del sistema informático a los diferentes usuarios. Sin que se afecte el tiempo de respuesta. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RNF04 |
| **Nombre del Requerimiento:** | Confiabilidad continúa del sistema. |
| **Características:** | La página web tendrá que estar en funcionamiento las 24 horas los 7 días de la semana. |
| **Descripción del requerimiento:** | La disponibilidad de la página debe ser continua con un nivel de servicio para los usuarios de 7 días por 24 horas, garantizando un esquema adecuado que permita la posible falla en cualquiera de sus componentes, contar con una contingencia, generación de alarmas. |
| **Prioridad del requerimiento:** Alta | |

|  |  |
| --- | --- |
| **Identificación del requerimiento:** | RNF05 |
| **Nombre del Requerimiento:** | Seguridad en información |
| **Características:** | El sistema garantizara a los usuarios una seguridad en cuanto a la información que se procede en el sistema. |
| **Descripción del requerimiento:** | Garantizar la seguridad del sistema con respecto a la información y datos que se manejan tales sean documentos, archivos y contraseñas. |
| **Prioridad del requerimiento:** Alta | |

## PLANTILLAS

|  |  |
| --- | --- |
| **Caso de uso** | Acceder al sistema, listar, mostrar detalles del socio. |
| **Iteración** | Modificación: 01 Sept. E. Mendoza. |
| **Actor principal** | Administrador. |
| **Objetivo en contexto** | Ver los registros de usuarios, saber qué usuarios interactuaron con el sistema cuando se realiza modificaciones en los datos. |
| **Precondiciones** | Deben obtenerse las identificaciones y claves de usuario apropiadas. |
| **Disparador** | El Administrar el sistema, desde su equipo (PC). |
| **Escenario** | 1. El Administrador se registra en el sistema como usuario.  2. El Administrador introduce su identificación de usuario.  3. El Administrador proporciona su clave.  4. El sistema despliega todos los botones de las funciones principales.  5. El Administrador selecciona “Socios” entre los botones de funciones principales.  6. El Administrador escoge “seleccionar Listar Socios”.  7. El sistema muestra a todos los socios registrados en el sistema.  8. El Administrador selecciona de la lista un socio.  9. El Administrador pulsa el botón “ver más”.  10. El sistema muestra la ventana de la vista del socio.  11. El sistema presenta al socio con los detalles de consumos, pagos, deudas, datos personales. |
| **Excepciones** | 1. La identificación o las claves son incorrectas  (Ver el caso de uso **Validar identificación**).  2. El Administrador selecciona “Listar todos los Socios (ver el caso de uso **Listar socios**).  4. No se dispone o no se ha configurado el sistema (se muestra el mensaje de error apropiado y ver el caso de uso **Configuraciones**). |
| **Prioridad:** | Moderada, se implementara después de las funciones básicas |
| **Cuándo estará disponible:** | En la primera etapa de la presentación. |
| **Frecuencia de uso** | Muchas veces por día |
| **Canal para el actor** | A través del sistema en PC. |
| **Actores secundarios** | Administrador del sistema |
| **Canales para los actores secundarios** | 1. Administrador del sistema  2. Servicios web |
| **Aspectos pendientes** | 1. ¿Qué mecanismos protegen el uso no autorizado de este sistema  por parte de los empleado?  2. ¿Es suficiente la seguridad? |

|  |  |
| --- | --- |
| **Caso de uso** | Mostrar reportes de socios |
| **Iteración** | Modificación: 01 de Sept. por M. Mendoza. |
| **Actor principal** | Administrador. |
| **Objetivo en contexto** | Ver los reportes de movimientos, por fechas de los socios dentro de la OTB. |
| **Precondiciones** | El sistema debe estar configurado por completo. |
| **Disparador** | El Administrador decide los tipos de reportes requiere en el sistema. |
| **Escenario** | 1. El Administrador ingresa al sistema con su identificación de usuario.  2. El sistema despliega todos los botones de las funciones principales  3. El Administrador selecciona “informes” entre los botones  4. El Administrador escoge “seleccionar Socios”.  5. El sistema muestra el reporte de Pagos, Lecturas, Consumos y estado  6. El Administrador selecciona un mes determinado  7. El Administrador pulsa el botón “Buscar”.  8. El sistema muestra el reporte de la fecha identificada.  9. El sistema presenta un reportes según fechas indicadas, socios, pagos, consumos, etc. |
| **Excepciones** | 1. La identificación o las claves son incorrectas o no se reconocen  2. La función de reporte por Socios “activo” o “ inactivo” no está configurada para este sistema  3. No se dispone o no se ha configurado el sistema con reportes  4. Se encuentra un reporte especifico |
| **Prioridad:** | Alta, por implementarse después de las funciones básicas. |
| **Cuándo estará disponible:** | En la segunda fase. |
| **Frecuencia de uso** | 5 veces por día |
| **Canal para el actor** | A través del sistema PC. |
| **Actores secundarios** | Administrador del sistema, equipo PC |
| **Canales para los actores secundarios** | 1. Administrador del sistema.  2. Equipo PC. |
| **Aspectos pendientes** | 1. ¿Qué tipo de permisos se manejan por parte de los empleados?  2. Es suficientemente confiable los reportes que genera? |

|  |  |
| --- | --- |
| **Caso de uso** | Acceder a Toma de lecturas |
| **Iteración** | Modificación: 02 de Sept. Por M. Mendoza. |
| **Actor principal** | Administrador. |
| **Objetivo en contexto** | Ver los reportes tomas de lecturas de medidores de socios dentro de la OTB. |
| **Precondiciones** | El sistema debe estar configurado por completo. |
| **Disparador** | El Administrador decide los tipos de reportes requiere en el sistema. |
| **Escenario** | 1. El Administrador ingresa al sistema con su identificación de usuario.  2. El sistema despliega todos los botones de las funciones principales  3. El Administrador selecciona “Lecturas” entre los botones  4. El Administrador escoge “seleccionar Socios”.  5. El sistema muestra el reporte de lecturas del socio  7. El Administrador selecciona un mes determinado  8. El Administrador pulsa el botón “Buscar”.  9. El sistema presenta una ventana en el cual debe registrar la lectura del socio.  10. Guardar cambios. |
| **Excepciones** | 1. La identificación o las claves son incorrectas o no se reconocen  2. No se dispone o no se ha configurado el sistema con reportes  3. Se encuentra un reporte especifico |
| **Prioridad:** | Alta, por implementarse después de las funciones básicas. |
| **Cuándo estará disponible:** | En la segunda fase. |
| **Frecuencia de uso** | 20 veces por día |
| **Canal para el actor** | A través del sistema PC. |
| **Actores secundarios** | Administrador del sistema, equipo PC |
| **Canales para los actores secundarios** | 1. Administrador del sistema.  2. Equipo PC. |
| **Aspectos pendientes** | 1. ¿Qué tipo de permisos se manejan por parte de los empleados? |

# CAPITULO IV. Elección de la metodología y herramientas

# CAPITULO V. Arquitectura del proyecto

# CAPITULO VI. Desarrollo del proyecto

# CAPITULO VII. Pruebas del proyecto

# CAPITULO VIII. Costo del proyecto

# Conclusiones y recomendaciones

# Bibliografía

# Apéndices

· Manual técnico

· Manual del usuario

· Manual de administración

· Pruebas de testeo

# Anexos