

**UNIVERSIDAD TECNOLÓGICA PRIVADA DE SANTA CRUZ**

**FACULTAD DE CIENCIAS Y TECNOLOGÍA**

**CARRERA: INGENIERÍA DE SISTEMAS**

**ÁREA: BASES DE DATOS Y SISTEMAS DE INFORMACIÓN**

**SISTEMA DE GESTIÓN ADMINISTRATIVO PARA LA URBANIZACIÓN “SACAGUAZU” EMPRESA “PENTA SOLUCIÓN”**

**AUTOR:** JOAQUIN EDGAR ALCON RUIZ

**EXAMEN DE GRADO PARA OPTAR EL TÍTULO DE LICENCIATURA EN INGENIERÍA EN SISTEMAS**

SANTA CRUZ – BOLIVIA 2021

**AGRADECIMIENTOS**

***A Dios***

*Por haberme dado la fortaleza, sabiduría y salud para poder cumplir todas mis metas y objetivos.*

***A mis padres, familiares y amigos.***

*Amparo y Edgar mis padres por cuidarme apoyarme en cada etapa de mi vida, por aconsejarme siempre con amor, sabiduría y siempre estar para mí cuando más los he necesitado sin ellos no sería nada hoy, mis dos grandes ejemplos y pilares.*

*A mis abuelos Adelma y Delman porque siempre me dieron su apoyo incondicional.*

*A mi abuelita que en paz descanse Luci que desde el cielo me cuida.*

***A mi universidad***

*Por haberme brindado todas las herramientas de aprendizaje en mis años de estudiante.*

*Todo este trabajo ha sido posible gracias a ustedes.*

ÍNDICE

[ASPECTOS GENERALES 1](#_Toc64880240)

[TÍTULO DEL PROYECTO 2](#_Toc64880241)

[1 INTRODUCCIÓN 2](#_Toc64880242)

[1.1. DEFINICIÓN DEL PROBLEMA 3](#_Toc64880243)

[1.2. SITUACIÓN DESEADA 3](#_Toc64880244)

[1.3. DELIMITACIONES 3](#_Toc64880245)

[1.3.1.- DELIMITACIÓN ESPACIAL 3](#_Toc64880246)

[1.3.2. DELIMITACIÓN TEMPORAL 3](#_Toc64880247)

[1.3.3. DELIMITACIÓN TECNOLÓGICA 3](#_Toc64880248)

[1.4. OBJETIVOS 4](#_Toc64880249)

[1.4.1.- OBJETIVO GENERAL 4](#_Toc64880250)

[1.4.2.- OBJETIVOS ESPECÍFICOS 4](#_Toc64880251)

[CAPITULO II 5](#_Toc64880252)

[2 MARCO TEÓRICO 5](#_Toc64880253)

[2.1. MARCO TEÓRICO SOBRE: BASE DE DATOS 6](#_Toc64880254)

[2.1.1. BASE DE DATOS 6](#_Toc64880255)

[2.1.2. SISTEMA GESTOR DE BASE DE DATOS (SGBD) 6](#_Toc64880256)

[2.1.3. CAMPO 6](#_Toc64880257)

[2.1.4. REGISTRO 6](#_Toc64880258)

[2.1.5. DATO 6](#_Toc64880259)

[2.1.6. DOMINIO 6](#_Toc64880260)

[2.1.7. INFORMACIÓN 6](#_Toc64880261)

[2.1.8. ADMINISTRADOR DE BASE DE DATOS (DBA): 6](#_Toc64880262)

[2.1.9. MODELO DE BASE DE DATOS 6](#_Toc64880263)

[2.1.10. FASES DEL DISEÑO DE LAS BASES DE DATOS 8](#_Toc64880264)

[2.1.11. PRIMARY KEY (PK) O LLAVE PRIMARIA 8](#_Toc64880265)

[2.1.12. FOREIGN KEY (FK) O LLAVE FORANEA 8](#_Toc64880266)

[2.1.13. INTEGRIDAD REFERENCIAL 9](#_Toc64880267)

[2.1.14. INTEGRIDAD DE ENTIDAD 9](#_Toc64880268)

[2.1.15. ATRIBUTOS 9](#_Toc64880269)

[2.1.16. NORMALIZACIÓN 9](#_Toc64880270)

[2.2 MARCO TEÓRICO SOBRE: METODOLOGÍA 10](#_Toc64880271)

[2.2.1 PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (PUDS) 10](#_Toc64880272)

[2.2.2 CARACTERÍSTICAS 10](#_Toc64880273)

[2.2.3 FASES 11](#_Toc64880274)

[2.2.4. UML 11](#_Toc64880275)

[2.2.5. DIAGRAMAS 11](#_Toc64880276)

[2.2.6. CLASIFICACIÓN 12](#_Toc64880277)

[2.2.7. DIAGRAMAS ESTÁTICOS 12](#_Toc64880278)

[2.2.8. DIAGRAMAS DINÁMICOS 12](#_Toc64880279)

[2.3 ARQUITECTURA DEL SISTEMA Y/O PATRONES DE DESARROLLO 14](#_Toc64880280)

[2.4.1 LENGUAJES 15](#_Toc64880281)

[2.5 PLATAFORMA 16](#_Toc64880282)

[2.6 BASE DE DATOS 16](#_Toc64880283)

[2.6.1 ¿QUÉ ES BASE DE DATOS? 16](#_Toc64880284)

[2.6.2¿QUÉ BASE DE DATOS SE UTILIZÓ? 16](#_Toc64880285)

[2.7 IDE 16](#_Toc64880286)

[2.7.1 ¿QUÉ ES UN IDE? 16](#_Toc64880287)

[2.7.2 ¿QUÉ IDE SE UTILIZÓ? 16](#_Toc64880288)

[CAPITULO III 17](#_Toc64880289)

[3 INGENIERIA DEL PROYECTO 17](#_Toc64880290)

[3.1. PLANIFICACION TEMPORAL 18](#_Toc64880291)

[3.1. CASO DE USO 20](#_Toc64880292)

[3.2. MODELO DE REQUISITOS 20](#_Toc64880293)

[3.2.1. REQUERIMIENTOS FUNCIONALES 20](#_Toc64880294)

[3.2.2. REQUERIMIENTOS NO FUNCIONALES 21](#_Toc64880295)

[3.2.3. DESCRIPCIÓN DE ACTORES 22](#_Toc64880296)

[3.3. MODELO DE ANÁLISIS 23](#_Toc64880297)

[3.3.1. DIAGRAMA GENERAL DE CASO DE USO 23](#_Toc64880298)

[3.4 MODELO DE DISEÑO 26](#_Toc64880299)

[3.4.1. DIAGRAMA DE SECUENCIA 26](#_Toc64880300)

[3.4.2. MODELO DE DOMINIO 28](#_Toc64880301)

[3.3.3 DIAGRAMA DE CLASES CONCEPTUALES 29](#_Toc64880302)

[3.3.4. NORMALIZACION 30](#_Toc64880303)

[3.4.5. DISEÑO DE LA BASE DE DATOS: LÓGICO Y FÍSICO 32](#_Toc64880304)

[3.4.6 DISEÑO DE REPORTES 33](#_Toc64880305)

[3.4.7. DERIVADOS DE SQL 34](#_Toc64880306)

[3.4.8 TRIGGERS 34](#_Toc64880307)

[3.4.9 CONSULTAS COMPLEJAS 35](#_Toc64880308)

[3.4.10 PROCEDIMIENTOS ALMACENADOS 36](#_Toc64880309)

[3.5 MODELO DE IMPLEMENTACIÓN 37](#_Toc64880310)

[3.5.1. MODELO DE COMPONENTES 37](#_Toc64880311)

[3.5.2. MODELO DE DESPLIEGUE 39](#_Toc64880312)

[3.5.3. IMPLEMENTACIÓN DE UN REQUERIMIENTO 42](#_Toc64880313)

[3.5.4. PLAN DE BACKUP DE LA BASE DE DATOS 45](#_Toc64880314)

[3.5 MODELO DE PRUEBAS 46](#_Toc64880315)

[3.5.1 PRUEBA FUNCIONAL 46](#_Toc64880316)

[CAPITULO IV 51](#_Toc64880317)

[4 CONCLUSIONES Y RECOMENDACIONES 51](#_Toc64880318)

[4.1 CONCLUSIONES 52](#_Toc64880319)

[4.2 RECOMENDACIONES 52](#_Toc64880320)

[BIBLIOGRAFÍA 53](#_Toc64880321)

[ANEXOS 54](#_Toc64880322)

**Índice de Figuras**

[**Figura 1:** Programación N capas 14](#_Toc64880323)

[**Figura 2:** Diagrama general de caso de uso 23](#_Toc64880324)

[**Figura 3:** Diagrama de secuencia sobre Registrar Clientes potenciales 26](#_Toc64880325)

[**Figura 4:** Diagrama de secuencia “Registrar Reserva” 27](#_Toc64880326)

[**Figura 5:** Modelo de dominio 28](#_Toc64880327)

[**Figura 6:** Diagrama de clases conceptual 29](#_Toc64880328)

[**Figura 7:** Modelo de la Base de Datos 32](#_Toc64880329)

[**Figura 8:**Diseño de reporte vendedores 33](#_Toc64880330)

[**Figura 9:**Trigger Log de resgistros de reserva 34](#_Toc64880331)

[**Figura 10:**Resultado del trigger 34](#_Toc64880332)

[**Figura 11**:Muestra los trabajadores y cuando han vendido por mes 35](#_Toc64880333)

[**Figura 12**:Procedimiento de almacenada(Muestra los lotes disponibles) 36](#_Toc64880334)

[**Figura 13:** Modelo de componente sobre el caso de uso de Registrar Clientes potenciales 37](#_Toc64880335)

[**Figura 14:** Modelo de componente sobre caso de Registrar Reserva 38](#_Toc64880336)

[**Figura 15:** Modelo de despliegue sobre el caso de Registrar Clientes potenciales 39](#_Toc64880337)

[**Figura 16:** Modelo de despliegue sobre el caso de Registrar Reserva 40](#_Toc64880338)

[**Figura 17:** Diseño de arquitectura física 41](#_Toc64880339)

[**Figura 18:** Diseño de login 42](#_Toc64880340)

[**Figura 19:** Diseño de Registro para Trabajadores, usuario y contraseña 43](#_Toc64880341)

[**Figura 20:** Menú del Administrador 44](#_Toc64880342)

[**Figura 21:** Realizar Reserva 44](#_Toc64880343)

[**Figura 22:** Registrar Cliente Potencial o Cliente 45](#_Toc64880344)

[**Figura 23:** Prueba funcional RegistroTrabajador Para login 46](#_Toc64880345)

[**Figura 24:** Prueba funcional RegistroTrabajador Para login (Guardado exitoso) 46](#_Toc64880346)

[**Figura 25:** Prueba funcional Registro Cliente 47](#_Toc64880347)

[**Figura 26:** Prueba funcional Registro Cliente(Registro exitoso) 47](#_Toc64880348)

[**Figura 27:** Prueba funcional Registro Cliente 48](#_Toc64880349)

[**Figura 28:** Prueba funcional Realizar Reserva 49](#_Toc64880350)

[**Figura 29:** Prueba funcional Realizar Reserva 50](#_Toc64880351)

[**Figura 30:** Prueba funcional Realizar Reserva 50](#_Toc64880352)

**Índice de Tablas**

[**Tabla 1:** Planificación temporal 18](#_Toc64880353)

[**Tabla 2:** Requerimientos funcionales 20](#_Toc64880354)

[**Tabla 3:** Requerimientos no funcionales 21](#_Toc64880355)

[**Tabla 4:** Descripción de los actores del sistema 22](#_Toc64880356)

[**Tabla 5:** Especificación de caso de uso Gestión Lotes/Terrenos(Registro Reserva) 24](#_Toc64880357)

[**Tabla 6:** Especificación de caso de uso Registrar Clientes potenciales 25](#_Toc64880358)

|  |  |
| --- | --- |
| UNIVERSIDAD TECNOLÓGICA PRIVADA DE SANTA CRUZ  SOCIEDAD ANÓNIMA | COD: PO-DFG-100-4 |
| **CASO DE ESTUDIO – EXAMEN DE GRADO** | VER: 2  VIGENTE: 05-07-2017 |



**Enunciado del caso de estudio**

ÁREA: Bases de datos y sistemas de información

CASO: SISTEMA DE GESTIÓN ADMINISTRATIVO PARA LA VENTA DE LOTES DE TERRENOS DE LA URBANIZACIÓN “SACAGUAZU” PARA LA EMPRESA “PENTA SOLUCIÓN”

Se necesita un sistema de información que gestione los terrenos que se tienen disponibles para la venta, los interesados en comprar un terreno, los clientes que compraron un lote y las cuotas de los mismos si es que lo están pagando a crédito.

La empresa de venta de terrenos “Penta Solución”, ha recibido los planos autorizados para la para la urbanización “SacaGuazu” y ahora tiene que vender los lotes de terreno que están disponibles.

Cada lote de terreno está definido por 4 puntos que conforman el tamaño y posición del lote. Cada punto está conformado por latitud y longitud (geoposicionamiento), ya que el diseño de cada terreno en la urbanización fue dibujado con ayuda de imágenes satelitales. Cuando una persona se acerca a preguntar por los lotes, es política de la empresa registrarlo como posible cliente, alguien a quien se le podrá ofrecer otros terrenos en el futuro. La persona que quiere comprar un lote puede hacerlo de dos maneras, al contado y al crédito. Cuando una persona compra el lote, este es asignado en el sistema con el nombre del propietario del lote dejando de estar disponible. El sistema debe ser capaz de mostrar un plano de los lotes indicando cuales están disponibles y cuáles no.

Cuando una persona compra un lote al crédito se generan de manera automática, como no pagadas, todas las cuotas asignadas a un mes y una gestión, que este deberá cancelar.

El sistema debe estar desarrollado para un entorno web, ya que hay varias formas de ofrecer lotes: desde el teléfono celular, computadora de escritorio o tableta, desde donde se puede mostrar la disponibilidad de terrenos. De manera adicional si una persona está interesada en un lote, pero todavía no se decide a comprarlo puede reservarlo con un pago de 100$. Esta reserva dura 2 meses. Después de este periodo la reserva queda anulada, el dinero de la reserva es devuelto al cliente y este tiene la opción de volver a reservarlo o no. Al ser consultado el encargado de sistemas, indico que lo más importante para poder iniciar la venta de los terrenos es poder mostrarlos en una interface gráfica. De esta manera se puede mostrar a los potenciales clientes los lotes que están disponibles, los reservados y los que ya han sido vendidos. Para poder hacer esto en el sistema ya deberían tener registrados los lotes con los 4 puntos cartesianos ya antes mencionados.

CAPÍTULO I

# ASPECTOS GENERALES

# TÍTULO DEL PROYECTO

Sistema de gestión administrativo para la urbanización “SACAGUAZU” empresa “PENTA SOLUCIÓN”

# 1 INTRODUCCIÓN

En los últimos años se ha desarrollado un crecimiento en las ciudades sin presentes, esto ha provocado el surgimiento de diferentes empresas dedicadas a la venta de terrenos en urbanizaciones. Dichas empresas dado que manejan diferentes tipos de documentos y gestionan también los pagos de sus terrenos, se han visto en la necesidad de buscar nuevas formas gestionar sus datos.

Es por esto que han optado por buscar la solución en tecnologías de Gestión Administrativos para un manejo más eficiente de los datos de dichas empresas. Estas tecnologías ya se han estado abriendo paso en el mercado de venta software, por la automatización y control que dichos programas pueden llegar a ofrecer.

Este es el caso de la empresa “PENTA SOLUCION” la cual está a adquirido una nueva urbanización llamada “SACAGUAZU” con esta nueva urbanización y cantidad de lotes a la venta, se busca poder implementar uno de estos sistemas a dicha empresa para poder automatizar y gestionar de mejor manera la venta de los terrenos.

Para el presente proyecto se desarrollará un sistema de gestión administrativo para la venta de lotes de terrenos en Santa Cruz, utilizando la metodología PUD “Proceso Unificado de Desarrollo” y UML “Lenguaje Unificado de Modelado” para el diseño de las tablas relacionales.

El equipo de desarrollo está conformado por 4 personas: Un analista de Sistemas y tres desarrolladores. El proyecto será efectuado en la ciudad de Santa Cruz de la Sierra, la misma que tendrá de fecha de inicio a partir del 17 de Febrero del año 2021 , el cual se tiene proyectado entregar el sistema en un tiempo de 6 meses. Para dicho proyecto se utilizará: Architect para los diseños y modelos de diagrama de datos, Visual code como IDE, PHP, JavaScripts, jquery, Ajax. Como lenguajes de programación. Html, Css como lenguajes de etiquetas, como lenguaje de base de datos SQL, como SGBD MySql.

## 1.1. DEFINICIÓN DEL PROBLEMA

La empresa “PENTA SOLUCIÓN” necesita un sistema de información de gestión. Por qué ha recibido los planos de autorización para la venta de los terrenos de la urbanización “SACAGUAZU”, para poder registrar a los interesados en comprar un terreno, los clientes que compraron un lote y las cuotas de los mismos si es que lo están pagando a crédito.

## 1.2. SITUACIÓN DESEADA

* Se requiere de un sistema que controle las solicitudes de los clientes de los servicios.
* Se desea que el sistema registre los clientes potenciales.
* Se requiere que el sistema muestre los lotes disponibles y su ubicación.
* Se requiere que se registren las reservas de los lotes.
* Se quiere que el sistema sea capaz de generar las cuotas con su plan de pago respectivo.
* Se requiere reportes para control administrativo de la empresa.

## 1.3. DELIMITACIONES

### 1.3.1.- DELIMITACIÓN ESPACIAL

El presente proyecto se realizará en el departamento Santa Cruz en la ciudad Santa Cruz de la Sierra.

### 1.3.2. DELIMITACIÓN TEMPORAL

El presente proyecto será desarrollado e implementado en un periodo de 6 meses de trabajo, días hábiles, a partir del 17 febrero del 2021 al 1 diciembre de 2021.

### 1.3.3. DELIMITACIÓN TECNOLÓGICA

Se aplicará los conocimientos del área de información, específicamente el análisis de Sistemas, conocimiento de programación, la metodología PUD, y Base de datos.

Las tecnologías a usar son:

* PUDS
* Programación orientada a objetos
* PHP
* JavaScript
* HTML
* Css
* MySql

## 1.4. OBJETIVOS

### 1.4.1.- OBJETIVO GENERAL

Desarrollar un sistema de gestión administrativo para la venta de lotes de terrenos de la urbanización “SACAGUAZU” Empresa “PENTA SOLUCION”.

### 1.4.2.- OBJETIVOS ESPECÍFICOS

Los objetivos específicos para lograr el objetivo general son los siguientes:

* Identificar y analizar los procesos de la institución “PENTA SOLUCION”.
* Identificar todos los requisitos funcionales en base a los procesos administrativos
* Realizar el modelo de requisitos, las capturas de los requisitos funcionales y no funcionales.
* Diseñar el modelo de datos para la implementación de la base de datos.
* Desarrollar la codificación de los puntos del sistema en base a los requisitos funcionales.
* Evaluar las pruebas de funcionamiento del sistema codificado.

# CAPITULO II

# 2 MARCO TEÓRICO

## 2.1. MARCO TEÓRICO SOBRE: BASE DE DATOS

### 2.1.1. BASE DE DATOS

Es aquella que está compuesta por un conjunto de datos almacenados y relacionados entre sí, permitiendo manipular los datos de una empresa. La función principal de una base de datos es mantener la integridad y seguridad de los datos ante cualquier incidente.

### 2.1.2. SISTEMA GESTOR DE BASE DE DATOS (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

### 2.1.3. CAMPO

Un campo es la mínima unidad de información a la que se puede acceder. Un campo o conjunto de ellos forma un registro

### 2.1.4. REGISTRO

También se llama tupla o Fila. Conjunto de campos lógicamente relacionados.

### 2.1.5. DATO

Es un valor de alguna característica de un objeto real sea concreto o imaginario.

### 2.1.6. DOMINIO

Es el conjunto de todos los posibles valores que puede tomar un atributo de la relación. No es más que un tipo de datos. Ej. Booleano, Entero, cadena de caracteres, etc.

### 2.1.7. INFORMACIÓN

Es un conjunto de datos convenientemente procesado, puede ser usado en la realización de cálculos y puede ser usado para toma de decisiones.

### 2.1.8. ADMINISTRADOR DE BASE DE DATOS (DBA):

Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos.

### 2.1.9. MODELO DE BASE DE DATOS

En función de la estructura utilizada para construir una base de datos, existen diversos modelos de bases de datos. El modelo de la base de datos define un paradigma de almacenamiento, estableciendo cómo se estructuran los datos y las relaciones entre estos. Las distintas operaciones sobre la base de datos (eliminación o sustitución de datos, lectura de datos, etc.) vienen condicionadas por esta estructura, y existen notables diferencias entre los principales modelos, cada uno de ellos con sus ventajas e inconvenientes particulares. Algunos de los más habituales son los siguientes:

#### 2.1.9.1. MODELO RELACIONAL

Constituyen el modelo de bases de datos más utilizado en la actualidad. Solucionan los problemas asociados a las bases de datos jerárquicas y en red, utilizando para ello un esquema basado en tablas, que resulta a la vez sencillo de comprender y fácil de utilizar para el análisis y la consulta de los datos. Las tablas contienen un número dado de registros (equivalentes a las filas en la tabla), así como campos (columnas), lo que da lugar a una correcta estructuración y un acceso eficiente.

#### 2.1.9.2. MODELO JERÁRQUICO

El modelo jerárquico organiza los datos en una estructura de árbol, en la que cada registro tiene un único elemento o raíz. Los registros del mismo nivel se clasifican en un orden específico. Ese orden se usa a manera de orden físico para almacenar la base de datos. El modelo es bueno para describir muchas relaciones del mundo real.

#### 2.1.9.3. MODELO DE RED

El modelo de red se basa en el modelo jerárquico, permitiendo relaciones de muchos a muchos entre registros vinculados, lo que implica registros principales múltiples. Basado en la teoría matemática de conjuntos, el modelo se construye con conjuntos de registros relacionados. Cada conjunto consiste de un registro propietario o principal y uno o más registros miembros o secundarios. Un registro puede ser miembro o secundario en múltiples conjuntos, permitiendo que este modelo represente relaciones complejas.

#### 2.1.9.4. MODELO ORIENTADO A OBJETOS

Este modelo define una base de datos como una colección de objetos, o elementos de software reutilizables, con funciones y métodos relacionados. Incorpora todo el concepto importante del paradigma de objetos:

Encapsulación

Herencia

Polimorfismo

#### 2.1.9.5. MODELO ENTIDAD-RELACIÓN

Este modelo capta las relaciones entre entidades del mundo real de forma muy similar al modelo de red, pero no está directamente ligado a una estructura física de la base de datos. En cambio, con frecuencia se lo usa para diseñar una base de datos conceptualmente.

Aquí, a las personas, lugares y cosas, acerca de las cuales se almacenan puntos de datos, se las denomina entidades, cada una de las cuales tiene ciertos atributos que en conjunto forman su dominio. La cardinalidad, o relaciones entre entidades, también se representan en diagramas.

### 2.1.10. FASES DEL DISEÑO DE LAS BASES DE DATOS

#### 2.1.10.1. DISEÑO CONCEPTUAL

El diseño conceptual incluye la creación de un esquema o modelo conceptual de la base de datos. Este modelo es independiente de las consideraciones físicas, incluyendo los sistemas de gestión de base de datos, los lenguajes de programación y las plataformas de hardware. Los usuarios no técnicos deben entender el esquema, así que no debería contener detalles sobre cómo se implementará la base de datos. Pero debe estar detallado en términos de la naturaleza, estructura y significado de los datos.

#### 2.1.10.2. DISEÑO LÓGICO

El objetivo del diseño lógico es transformar el esquema genérico y conceptual en un modelo de datos determinado para un sistema de gestión de bases de datos determinado. El diseño lógico puede realizarse manualmente, o automáticamente en algunos casos, a través del uso de herramientas CASE (ingeniería de software asistida por computadora) desde un diseño conceptual. En cualquier caso, el resultado final es un conjunto de comandos de lenguaje de definición de datos, que puede ser usado de forma interactiva, o como parte de un programa de computadora para crear la base de datos.

#### 2.1.10.3. DISEÑO FÍSICO

El diseño físico es el proceso de implementación física del modelo de datos lógico en un sistema de gestión de bases de datos. Incluye escoger las estructuras de archivos determinadas en las que almacenar las tablas, o relaciones, de la base de datos, y en asegurar que se puede acceder a las relaciones de forma rápida, eficiente y segura. Los diseñadores de bases de datos pueden necesitar pensar sobre los tiempos de acceso a la base de datos para las transacciones que se usan habitualmente, así como el número medio de transacciones procesadas por minuto y la cantidad de espacio que necesitará la base de datos. Un índice en unos campos, o columnas, determinados en una base de datos puede mejorar los tiempos de acceso, pero el diseñador de bases de datos debe decidir qué campos indexar.

### 2.1.11. PRIMARY KEY (PK) O LLAVE PRIMARIA

Se llama primary key o llave primaria a un campo o una combinación de campos que identifica de forma única a cada fila de una tabla. No puede haber dos filas en una tabla que tengan la misma clave primaria.

La selección de una clave primaria es muy importante en el diseño de una BD ya que es un elemento clave en la unión de tablas y el concepto total de una BD.

Las llaves primarias deben ser únicas y no nulas (NOT NULL).

### 2.1.12. FOREIGN KEY (FK) O LLAVE FORANEA

Una llave foránea, externa o ajena (Foreign Key) es un campo de una tabla “X” que sirve para enlazar o relacionar entre sí con otra tabla “Y” en la cual el campo de esta tabla es una llave primaria (Primary Key).

Para que sea una llave foránea un campo, esta tiene que ser una llave primaria en otra tabla.

### 2.1.13. INTEGRIDAD REFERENCIAL

La integridad referencial es una propiedad de la base de datos. La misma significa que la clave externa de una tabla de referencia siempre debe aludir a una fila válida de la tabla a la que se haga referencia. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones de actualización y eliminación.

### 2.1.14. INTEGRIDAD DE ENTIDAD

Pretende que cada entidad que se guarda en la base de datos sea identificable de un modo único, es decir, que evitemos la información redundante. La integridad de entidad define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.

### 2.1.15. ATRIBUTOS

Los atributos son las columnas de una relación y describen características particulares de ella.

Existen varios tipos de atributos que son:

#### 2.1.15.1. ATRIBUTOS MONOVALUADOS

Poseen un solo valor en particular Eje.: edad, sueldo, marca.

#### 2.1.15.2. ATRIBUTOS MULTIVALUADOS

Pueden poseer varios valores para una entidad Eje.: Email, oficios, condecoraciones, teléfonos.

#### 2.1.15.3. ATRIBUTOS OBLIGATORIO

Siempre tienen un valor asignado Eje.: fecha de nacimiento, carrera, marca, precio.

#### 2.1.15.4. ATRIBUTOS OPCIONALES

Pueden registrarse o no de la base de datos Eje.: religión, Partido político, etc.

#### 2.1.15.5. ATRIBUTOS CALCULABLES O DERIVADOS

Se dice calculable porque su valor se puede obtener de otros atributos almacenados en la base de datos.

### 2.1.16. NORMALIZACIÓN

El proceso de normalización de una base de datos relacional consiste en aplicar una serie de reglas para evitar a futuro realizar o consultas innecesariamente complejas. En otras palabras, están enfocadas en eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas. Las bases de datos se normalizan para:

* Evitar la redundancia de datos.
* Proteger la integridad de los datos.
* Evitar problemas de actualización de los datos en las tablas.

#### 2.1.16.1. PRIMERA FORMA NORMAL

Se considera que una relación está en primera forma normal cuando todos sus elementos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles. Definiciones:

* No debe haber tuplas repetidas.
* No debe importar el orden de las tuplas.
* Existencia de una llave primaria.
* Atributos Atómicos.

#### 2.1.16.2. SEGUNDA FORMA NORMAL

Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir, que no existen dependencias parciales. Todos los atributos que no son clave principal deben depender únicamente de la clave principal.

#### 2.1.16.3. TERCERA FORMA NORMAL

La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva en los atributos que no son clave.

## 2.2 MARCO TEÓRICO SOBRE: METODOLOGÍA

### 2.2.1 PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (PUDS)

Es un conjunto de actividades necesarias para transformar los requisitos de usuario en un producto software. Es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura, y por ser iterativo e incremental.

**Fuente:** (I., G., & Rumbaugh, 2000)

### 2.2.2 CARACTERÍSTICAS

#### 2.2.2.1 DIRIGIDO POR CASOS DE USO

Un caso de uso es una pequeña funcionalidad del sistema que devuelve al usuario un resultado importante. Representa la unidad atómica funcional a través del cual se realizan todas las actividades necesarias para solucionar un problema, el conjunto de casos de uso de un sistema representa la funcionalidad total del producto. Los casos de uso no solo inician el proceso de desarrollo, sino que brindan un hilo constructor a través de todo el proceso, los casos se analizan, diseñan se implementan y se construyen los modelos de prueba.

#### 2.2.2.2 CENTRADO EN LA ARQUITECTURA

El Proceso Unificado, asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos, y vistas que definen la arquitectura de software de un sistema.

La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería. Etc.

La arquitectura del sistema, es un modelo que permite ver al producto antes de que este haya sido construido. Es decir, los casos de uso representan la funcionalidad del sistema y la arquitectura la forma.

#### 2.2.2.3 ITERATIVO E INCREMENTAL

El proceso Unificado, es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas: Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones.

Es práctico dividir al proyecto en pequeños mini proyectos. Estos mini proyectos o iteraciones incrementan la funcionalidad del sistema, para cada mini proyecto se realizan 5 flujos de trabajo: requisitos, análisis, diseño, implementación y prueba.

**Fuente:** (I., G., & Rumbaugh, 2000)

### 2.2.3 FASES

#### 2.2.3.1. INICIO

En la fase de inicio se desarrolla una descripción del producto final, y se representa el análisis del negocio. En esta fase se identifican las principales funciones del sistema.

#### 2.2.3.2. ELABORACIÓN

Durante esta fase se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura, la resolución de riesgos altos, nuevos requisitos y se ajustan estimaciones.

#### 2.2.3.3. CONSTRUCCIÓN

El producto crece hasta convertirse en un sistema completo preparado, para ser empleado a la comunidad de usuarios, puede que no esté completamente libre de defectos. Muchos de esos defectos se descubrirán y solucionarán durante la fase de transición.

#### 2.2.3.4. TRANSICIÓN

Esta fase cubre el periodo de entrega del producto a la comunidad de usuarios. Se capacita al personal o usuario finales de la aplicación, se crean los manuales o ayudas del sistema.

**Fuente:** (I., G., & Rumbaugh, 2000)

### 2.2.4. UML

El Lenguaje Unificado de Modelado (UML - Unified Modeling Lenguaje) es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

### 2.2.5. DIAGRAMAS

Un diagrama es una representación gráfica de un conjunto de elementos, se utilizan para ver al sistema de diferentes perspectivas. Como ningún sistema puede ser comprendido completamente desde una única perspectiva, UML utiliza 9 diagramas que permiten centrase en diferentes aspectos del sistema independientemente.

### 2.2.6. CLASIFICACIÓN

Se dispone de dos tipos diferentes de diagramas, los que dan una vista estática del sistema y los que dan una visión dinámica.

### 2.2.7. DIAGRAMAS ESTÁTICOS

#### 2.2.7.1. DIAGRAMA DE CLASES

Este diagrama sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido.

#### 2.2.7.2. DIAGRAMA DE OBJETO

Un diagrama de objeto, es esencialmente una instancia de un diagrama de clase o la parte estática de un diagrama de interacción, en un diagrama de objeto podremos encontrar objetos y enlaces.

#### 2.2.7.3. DIAGRAMA DE COMPONENTES

Un diagrama de componentes muestra la relación estructural de los componentes de un sistema de software. Los componentes se comunican entre sí mediante interfaces.

#### 2.2.7.4. DIAGRAMA DE DESPLIEGUE

Los Diagramas de Implementación se usan, para modelar sólo componentes que existen como entidades en tiempo de ejecución; no se usan para modelar componentes solo de tiempo de compilación o de tiempo de enlazado.

### 2.2.8. DIAGRAMAS DINÁMICOS

#### 2.2.8.1. DIAGRAMA DE CASOS DE USO

Los diagramas de casos de uso, ofrecen una visión general de los actores involucrados en un sistema, las diferentes funciones que necesitan esos actores y cómo interactúan estas diferentes funciones.

#### 2.2.8.2. DIAGRAMA DE SECUENCIA

Los diagramas de secuencia, muestran como los objetos interactúan entre si y el orden en que se producen esas interacciones. Es importante tener en cuenta que muestran las interacciones para un escenario en particular.

**2.2.8.3. DIAGRAMA DE ACTIVIDAD**

Los diagramas de actividad, se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema.

**2.2.8.4. DIAGRAMA DE COLABORACIÓN**

Este diagrama muestra la interacción entre varios objetos, y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencia, un diagrama de colaboración muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los diagramas de secuencia y los diagramas de colaboración expresan información similar, pero en una forma diferente.

**2.2.8.5. DIAGRAMA DE ESTADO**

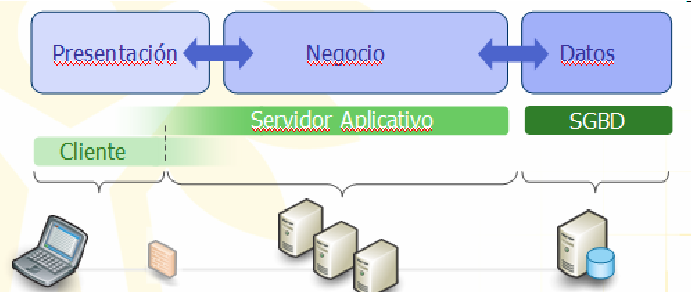
Representan la secuencia de estados, por los que un objeto o una interacción entre objetos pasan durante su tiempo de vida en respuesta a estímulos recibidos.

## 2.3 ARQUITECTURA DEL SISTEMA Y/O PATRONES DE DESARROLLO

Se utilizará la programación por capas ya que es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

**Figura 1:** Programación N capas



**Fuente:** Elaboración propia

#### 2.3.1.1. CAPA PRESENTACIÓN

En esta capa se crea la interfaz del usuario. Su única función es pasarle las acciones que realice el usuario a la capa de negocio.

#### 2.3.1.2. CAPA LÓGICA

Se denomina capa lógica (e incluso capa negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él.

#### 2.3.1.3. CAPA DATOS

Esta capa se encarga de guardar los datos. Será donde se gestione todo lo relativo a la base de datos y a la creación, edición y borrado de datos de ésta.

### 2.4.1 LENGUAJES

#### 2.4.1.1 ¿QUÉ ES UN LENGUAJE DE PROGRAMACIÓN?

En informática, se conoce como lenguaje de programación a un programa destinado a la construcción de otros programas informáticos. Su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que serán luego llevados a cabo por un ordenador o sistema informático, permitiendo controlar así su comportamiento físico, lógico y su comunicación con el usuario humano.

Fuente: (Raffino, 2018)

#### 2.4.1.2 ¿QUÉ LENGUAJE SE UTILIZÓ?

En el sistema que hemos desarrollado se utilizó 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. También se utilizará JavaScripts, SQL.

#### 2.4.1.3¿POR QUÉ SE UTILIZÓ PHP?

PHP es un lenguaje totalmente libre y abierto. Tiene una curva de aprendizaje muy baja, su sintaxis es simple y cumple estándares básicos de la programación orientada a objetos. No son necesarios complejos entornos de desarrollo, que incluso necesitan su propio periodo de aprendizaje.

#### 2.4.1.4 ¿PORQUE JAVASCRIPT?

Javascript es un lenguaje que puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos.

#### 2.4.1.5 ¿PORQUE AJAX?

Ajax se define como una técnica para el desarrollo de páginas (sitios) web que implementan aplicaciones interactivas, es decir que no es necesario recargar la página para intercambiar datos entre el servidor y el cliente.

#### 2.4.1.6 ¿PORQUE JQUERY?

Es una librería de JavaScript, esta librería de código abierto, simplifica la tarea de programar en JavaScript, es decir ahorra código a la hora de programar.

#### 2.4.1.7 ¿QUÉ ES SQL?

El Lenguaje de Consulta Estructurado popularmente conocido por sus siglas en inglés como SQL, es un tipo de lenguaje de programación que ayuda a solucionar problemas específicos o relacionados con la definición, manipulación e integridad de la información representada por los datos que se almacenan en las bases de datos.

## 2.5 PLATAFORMA

Se decidió desarrollar la aplicación para plataformas webs para que se pueda correr desde navegadores: Chrome, Mozilla y Opera.

## 2.6 BASE DE DATOS

### 2.6.1 ¿QUÉ ES BASE DE DATOS?

Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada.

### 2.6.2¿QUÉ BASE DE DATOS SE UTILIZÓ?

Se decidió utilizar MySql porque es una avanzada base de datos relacional de código abierto que soporta SQL (RELACIONAL).

## 2.7 IDE

### 2.7.1 ¿QUÉ ES UN IDE?

Es un Entorno de Desarrollo Integrado que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor compilador de código y un depurador.

### 2.7.2 ¿QUÉ IDE SE UTILIZÓ?

Visual studio code. Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git.

# CAPITULO III

# 3 INGENIERIA DEL PROYECTO

## 3.1. PLANIFICACION TEMPORAL

La planificación temporal tiene como objetivo evitar el retraso en la entrada del sistema. Un sistema de software general debe cumplir plazos o fechas de entrega impuestas por departamentos fuera del equipo de desarrollo en consecuencia pueden ser fechas irrealistas.

(Almiron, 2018)

**Tabla 1:** Planificación temporal

|  |  |  |  |
| --- | --- | --- | --- |
| **SISTEMA DE GESTIÓN ADMINISTRATIVO PARA LA VENTA DE LOTES DE TERRENOS DE LA URBANIZACIÓN “SACAGUAZU” PARA LA EMPRESA “PENTA SOLUCIÓN”** | | | |
|
| **TAREA** | **DURACIÓN** | **DESDE** | **HASTA** |
| **Fase de Inicio** | **11 días** | 17 de Febrero de 2021 | De 3 Marzo de 2021 |
| Captura de requisitos | 6 días | 17 de Febrero 2021 | 24 de Febrero de 2021 |
| Estimación de la duración | 5 días | 25 de marzo de 2020 | 03 de marzo de 2021 |
| **Fase de Elaboracion** | **15 días** | 4 de Marzo de 2021 | 24 de Marzo de 2021 |
| Identificación de los casos de uso | 3 días | 4 de Marzo de 2021 | 8 de marzo de 2021 |
| Diagrama General de casos de uso | 3 días | 9 de marzo de 2021 | 11de Marzo de 2021 |
| Diagrama de clases conceptuales | 3 días | 12 de Marzo de 2021 | 16 de Marzo de 2021 |
| Diagrama de componente | 3 días | 17 de Marzo de 2021 | 19 Marzo de 2021 |
| Diagrama de despliegue | 3 días | 22 Marzo de 2021 | 24 de Marzo 2021 |
| **Fase de Construccion** | **100 días** | 25 de Marzo de 2021 | 08 de Septiembre de 2021 |
| Gestión Lotes / Terrenos, Lotes/ Terrenos disponibles, Urbanización. | 20 días | 25 de Marzo de 2021 | 21 de abril de 2021 |
| Gestión lotes/terrenos vendidos, clientes, clientes potenciales | 20 días | 22 de abril de 2021 | 16 de junio de 2021 |
| Generación de cuotas | 20 días | 17 de Junio de 2021 | 14 de Julio de 2021 |
| Gestión Generación plan de pago, | 20 días | 15 de Julio de 2021 | 11 de Agosto de 2021 |
| Reportes de lotes vendidos, lotes y cuotas, resultado de vendedores | 20 días | 12 de Agosto de 2021 | 8 de Septiembre de 2021 |
| **Fase de Transición** | **60 días** | 9 de Septiembre de 2021 | 26 de junio de 2020 |
| Manual de Usuario. | 20 días | 9 de Septiembre de 2021 | 6 de Octubre de 2021 |
| Acta de capacitación al Usuario | 20 días | 7 de Octubre de 2021 | 3 de Noviembre de 2021 |
| Elaboración de manuales | 20 días | 4 de Noviembre de 2021 | 1 Diciembre de 2021 |
| **TOTAL DÍAS** | **186** | 17 de Febrero de 2021 | 1 Diciembre de 2021 |
|

**Fuente:** Elaboración Propia

## 3.1. CASO DE USO

Los siguientes casos de uso a tomar en cuenta para el modelado serán, gestión Clientes, Gestion Rerva de Lotes.

## 3.2. MODELO DE REQUISITOS

### 3.2.1. REQUERIMIENTOS FUNCIONALES

**Tabla 2:** Requerimientos funcionales

|  |  |  |
| --- | --- | --- |
| **NRO** | **REQUERIMIENTO** | **DESCRIPCIÓN** |
| RF1 | Gestión Lotes/Terrenos | Administrar la información de los lotes/terrenos, así como crear, modificar o eliminar. |
| RF2 | Gestión Lotes/Terrenos disponibles | Administrar la información de los lotes/terrenos disponibles, así como modificar su estado. |
|  | Gestión Urbanización | Administrar la información de las urbanizaciones, así como crear, modificar o eliminar. |
| RF3 | Gestión Lotes/Terrenos Vendidos. | Administrar la información de los lotes/terrenos vendidos. |
| RF4 | Gestión Clientes | Administrar la información de los Clientes así como crear, modificar o eliminar. |
| RF5 | Gestión Clientes Potenciales | Administrar la información de todos los clientes potenciales , así como crear, modificar o eliminar. |
| RF6 | Gestión Contrato |  |
| RF7 | Gestionar Venta de Lotes/terrenos | Administrar la información de todas las ventas de lotes/terrenos, así como crear, modificar. |
| RF8 | Gestionar Generación de Plan de pago | Administrar la información necesaria para la generación de plan pago, así como crear, modificar. |
| RF9 | Gestión de Cuotas para pago de terrenos | Registrar la información del pago de cada cuota generada, por lote/terreno. |
| RF10 | Gestión Reserva | Administrar la información de las reservas de los lotes/terrenos, así como crear, modificar. |
| RF11 | Gestión Persona Jurídica | Administrar la información de las de la persona jurídica de los lotes/terrenos, así como crear y modificar. |
| RF12 | Reporte de lotes vendidos | Poder mostrar los lotes vendidos de fechas determinadas. |
| RF13 | Reporte de lotes y cuotas | Visualización y reportes de los lotes que con más cuotas pagadas |
| RF14 | Reporte de vendedores | Se visualizara cuantos lotes vendió cada vendedor. |

**Fuente:** Elaboración Propia

### 3.2.2. REQUERIMIENTOS NO FUNCIONALES

**Tabla 3:** Requerimientos no funcionales

|  |  |  |
| --- | --- | --- |
| **NRO** | **REQUERIMIENTO** | **DESCRIPCIÓN** |
| RNF1 | Interfaz del sistema | * La interfaz de usuario deberá ser desarrollada para un entorno web. * El manejo de la interfaz debe ser intuitivo para el usuario de manera realice sus actividades en el sistema, de manera efectiva. |
| RNF2 | Desempeño del sistema | * El sistema debe estar disponible las 24 horas. * El sistema debe contar con validaciones en los formularios. |
| RNF3 | Estructura del sistema | * El motor de base de datos debe ser MySql. * El lenguaje de programación a utilizar es PHP, JavaScript |
| RNF4 | Seguridad | * El sistema deberá permitir que los usuarios que tengan acceso al sistema se autentifiquen antes de iniciar su sesión. |
| RNF5 | Backups | * Permitir la realización manual y automática de backup. |

**Fuente:** Elaboración Propia

### 3.2.3. DESCRIPCIÓN DE ACTORES

**Tabla 4:** Descripción de los actores del sistema

|  |  |  |
| --- | --- | --- |
| **ACTOR** | **DESCRIPCIÓN** | **OBJETIVOS** |
| Administrador | Actor que se encarga de la gestión de todos los datos relacionados a usuarios del sistema | * Gestión Lotes/Terrenos Gestionar Permisos * Gestión Urbanización * Gestión Clientes Potenciales * Gestión de Cuotas para pago de terrenos * Gestión Persona Jurídica * Reporte de lotes vendidos * Reporte de lotes y cuotas |
| Vendedor | Se encarga de registrar a los clientes, las ventas de los terrenos y las reservas. | * Gestión Lotes/Terrenos disponibles * Gestión Clientes * Gestionar Venta de Lotes/terrenos * Gestión Reserva * Gestionar Generación de Plan de pago |
| Cajero | Actor que es encargado de realizar los cobros de los planes de pago y sus respectivas cuotas. | * Gestión de Cuotas para pago de terrenos |

**Fuente:** Elaboración Propia

## 3.3. MODELO DE ANÁLISIS

### 3.3.1. DIAGRAMA GENERAL DE CASO DE USO

**Figura 2:** Diagrama general de caso de uso



**Fuente:** Elaboración Propia

3.3.2. ESPECIFICACIÓN DE CASOS DE USO SELECCIONADOS

#### 3.3.2.1. Caso de uso “Gestión Lotes/Terrenos”

**Tabla 5:** Especificación de caso de uso Gestión Lotes/Terrenos(Registro Reserva)

|  |  |  |
| --- | --- | --- |
| ESPECIFICACIÓN DE REQUISITOS | | |
| Nombre: | Interfaz de visualización Gestión Lotes/Terrenos | |
| Descripción: | Muestra una interfaz para que el usuario pueda realizar Reserva de lotes | |
| Dependencias: | Los lotes tiene que estar disponibles para poder realizar la reserva | |
| Actores: | Administrador, vendedor | |
| Flujo Normal | Actor | Sistema |
|  | 1.- El actor solicita ingreso al sistema | 2.- El sistema muestra un formulario de login para autentificación. |
| 3.- El actor ingresa su usuario y contraseña | 4.- El Sistema valida los campos ingresados. |
| 5.- El actor va a la opción de gestionar Lotes y terrenos | 6.- El sistema muestra la interface de listado los terrenos disponibles y da la opción de crear nueva reserva |
| 7.- El actor hace clic en Ingresar nueva reserva | 8.- El sistema muestra la interfaz de registrar Reserva |
|  | 9.- El sistema carga los datos |
| 10.- El actor presiona el botón de registrar. | 11.- El sistema procesa los datos para respectivo guardado de las solicitud. |
|  | 12.- El sistema le mostrar un mensaje de guardado correctamente. |
| Flujo alterno |  |  |
|  | 3.1 si el actor ingresa mal su usuario o contraseña el sistema muestra un mensaje de “usuario incorrecto |  |
|  | 11.1.- El sistema alertara al usuario si hay campos obligatorios por rellenar y así validar el guardado. |
|  | 12.1.- El sistema lanzara error si hubiera al momento de guardar los datos. |

**Fuente:** Elaboración Propia

#### 3.3.2.2. Caso de uso “Registrar Clientes potenciales”

**Tabla 6:** Especificación de caso de uso Registrar Clientes potenciales

|  |  |  |
| --- | --- | --- |
| ESPECIFICACIÓN DE REQUISITOS | | |
| Nombre: | Interfaz de visualización Registrar Clientes potenciales | |
| Descripción: | Muestra una interfaz para que el usuario pueda realizar el registro de los clientes y clientes potenciales. | |
| Dependencias: |  | |
| Actores: | Vendedor | |
| Flujo Normal | Actor | Sistema |
|  | 1.- El actor solicita ingreso al sistema | 2.- El sistema muestra un formulario de login para autentificación. |
| 3.- El actor ingresa su usuario y contraseña | 4.- El Sistema valida los campos ingresados. |
| 5.- El actor va a la opción de Registrar nuevo cliente | 6.- El sistema muestra la interface de registro de nuevos clientes |
| 7.- El actor hace clic en registrar Cliente | 8.- El sistema muestra la interfaz de registra a los clientes nuevos. |
|  | 9.- El sistema Muestra los clientes potenciales. |
|  | 10.- El sistema muestra un mensaje de Guardado exitoso. |
| Flujo alterno |  |  |
|  | 3.1 si el actor ingresa mal su usuario o contraseña el sistema muestra un mensaje de “usuario incorrecto |  |
|  | 14.1.- El sistema alertara al usuario si hay campos obligatorios por rellenar y así validar el guardado. |
|  | 15.1.- El sistema lanzara error si hubiera al momento de guardar los datos. |

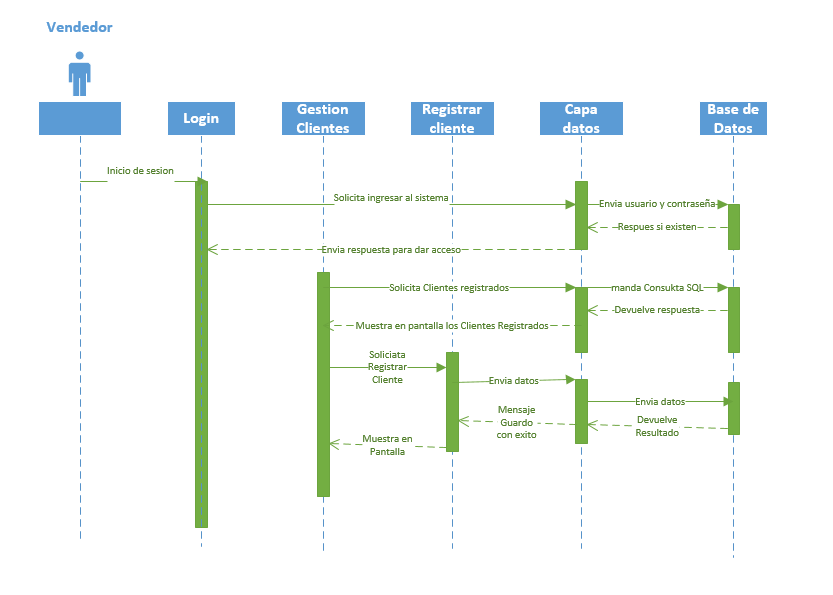
**Fuente:** Elaboración Propia

## 3.4 MODELO DE DISEÑO

### 3.4.1. DIAGRAMA DE SECUENCIA

#### 3.4.1.1. REGISTRAR CLIENTES POTENCIALES

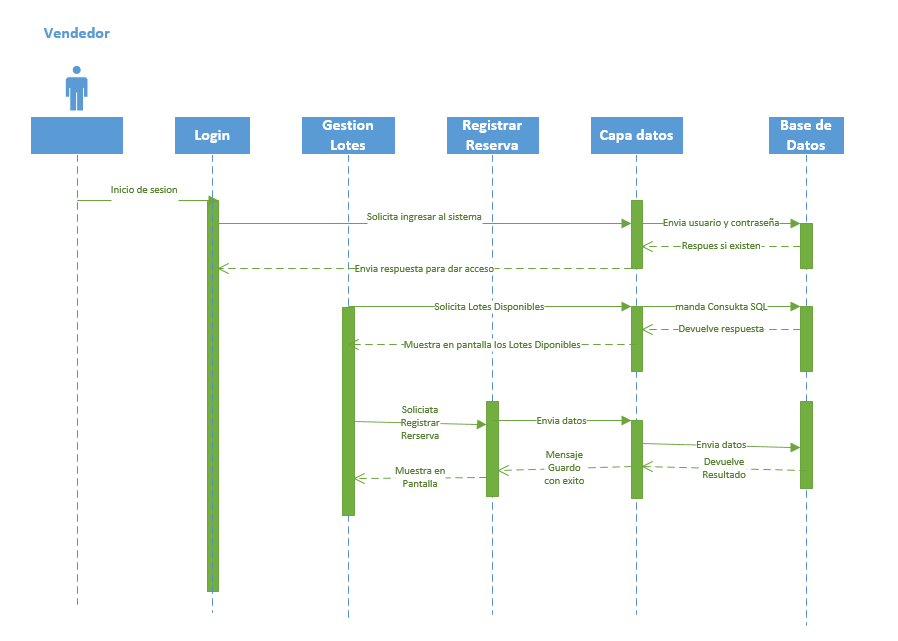
**Figura 3:** Diagrama de secuencia sobre Registrar Clientes potenciales



**Fuente:** Elaboración Propia

#### 3.4.1.2. REGISTRAR DETALLE MATERIAL (MODIFICAR)

**Figura 4:** Diagrama de secuencia “Registrar Reserva”



**Fuente:** Elaboración Propia

### 3.4.2. MODELO DE DOMINIO

**Figura 5:** Modelo de dominio



**Fuente:** Elaboración Propia

### 3.3.3 DIAGRAMA DE CLASES CONCEPTUALES

**Figura 6:** Diagrama de clases conceptual



**Fuente:** Elaboración Propia

### 3.3.4. NORMALIZACION

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Persona** | ci | Primer\_Nombre | Segundo\_Nombre | Apellido\_Paterno | Apellido\_Materno | Telefono\_Celular | Telefono\_Fijo |

PK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Trabajador** | Id\_Trabajador | Fecha\_Inicio | Rol | Usuario | Contraseha | Ci\_Identidad |

PK FK FK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cliente** | Id\_Cliente | Fecha\_Registro | TipoCliente | Ci\_Identidad |

PK FK FK

|  |  |  |  |
| --- | --- | --- | --- |
| **Persona\_Juridica** | Id\_Persona\_Juridica | Comentario | Ci\_Identidad |

PK FK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Urbanizacion** | Id\_Urbanizacion | Ciudad | ExtencionMTS2 | Nombre\_Urbanizacion | Id\_persona\_Juridica |

PK FK

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Lotes** | Id\_Lotes | Estado | Latitud1 | Latitud2 | Latitud3 | Latitud4 | Longitud1 | Longitud2 | Longitud3 | Longitud4 | Numero\_Lote | Precio | TamanhoMts2 | Tipo\_Moneda | Id\_urbanizacion |

PK FK FK

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Lote\_Vendido** | Id\_Lote\_Vendido | Fecha\_Vente | Ci\_Cliente | Nombre\_Cliente | Id\_Cliente | Id\_Lotes | estadocuota |

PK FK FK FK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Plan\_de\_Pago** | Id\_Plan\_Pago | Fecha\_Inicio | Fecha\_Fin | Gestion\_Deuda | Monto\_Total | Tipo\_Moneda | Tipo\_Plan\_Pag | Id\_Lote\_Vendido | Id\_Cliente |

PK FK FK FK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cuota** | Id\_Cuota | Fecha\_Cuota | Estado | Monto\_Cuota | Monto\_Restante | Monto\_Total\_Pagar | Id\_Cliente | Id\_Plan\_Pago | Id\_Trabajador |

PK PK FK FK FK

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Reserva** | Id\_Reserva | Estado | Fecha\_inicio | Fecha\_Fin | Monto\_Reserva | Tipo\_Moneda | Id\_Lote | Id\_Cliente |

PK FK FK FK

|  |  |  |
| --- | --- | --- |
| **estadocuota** | Id\_Estadocuota | Nombre |

PK

|  |  |  |
| --- | --- | --- |
| **estadoreserva** | Id\_EstadoReserva | Nombre |

PK

|  |  |  |
| --- | --- | --- |
| **estadolote** | Id\_EstadoLote | Nombre |

PK

|  |  |  |
| --- | --- | --- |
| **roles** | Id\_Role | Nombre\_Role |

PK

|  |  |  |
| --- | --- | --- |
| **tipo\_pago** | Id\_Tipo\_Pago | Nombre |

PK

|  |  |  |
| --- | --- | --- |
| **Tipo\_Cliente** | Id\_Tipo\_Cliente | Nombre |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Contrato** | Id\_contrato | Estado | Id\_Plan\_pago | Id\_Cliente | Fecha\_Contrato |

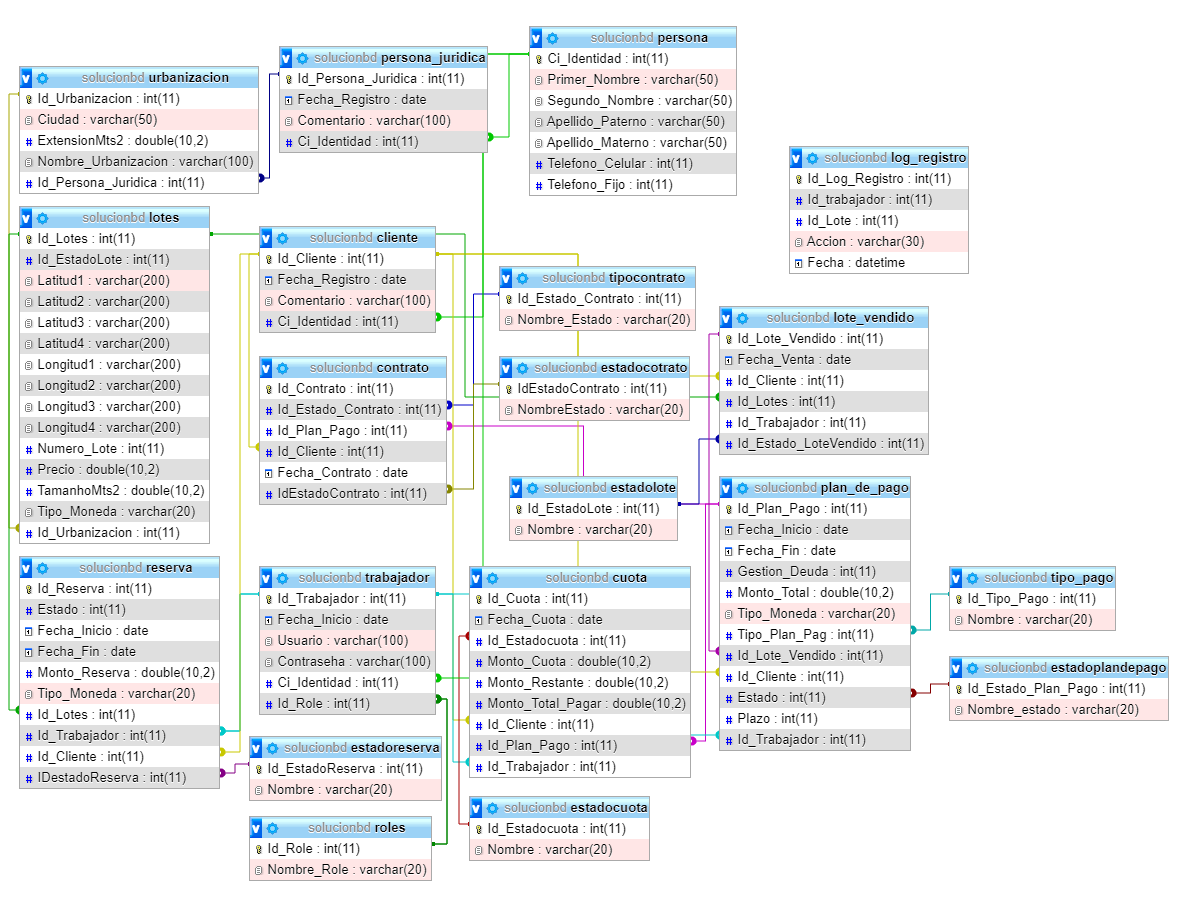
PK FK FK FK

|  |  |  |
| --- | --- | --- |
| **Tipo\_EstadoContrato** | IdEstadoContrato | Nombre\_Estado |

PK

### 3.4.5. DISEÑO DE LA BASE DE DATOS: LÓGICO Y FÍSICO

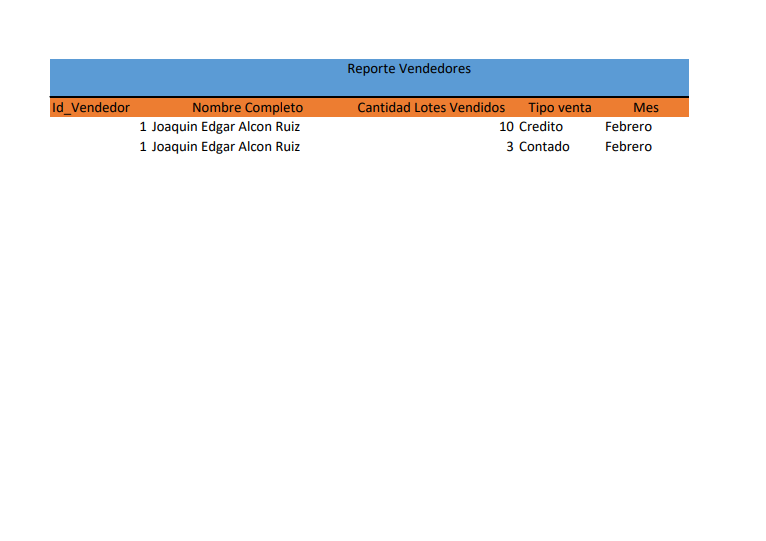
**Figura 7:** Modelo de la Base de Datos



**Fuente:** Elaboración Propia

### 3.4.6 **DISEÑO DE REPORTES**

**Figura 8:**Diseño de reporte vendedores



**Fuente:** Elaboración Propia

### 3.4.7. DERIVADOS DE SQL

### 3.4.8 TRIGGERS

**Figura 9:**Trigger Log de resgistros de reserva

|  |
| --- |
| CREATE TRIGGER `LogRegistroReserva` AFTER INSERT ON `reserva` FOR EACH ROW BEGIN  INSERT INTO log\_registro (Id\_trabajador,Id\_Lote,Accion,Fecha)  VALUES (NEW.Id\_trabajador,NEW.Id\_Lotes,'Se inserto lote',NOW());  END  $$  DELIMITER ; |

**Fuente:** Elaboración Propia

Este trigger se dispara cada que se Registra una reserva de lote, Guarda al trabajador que lo hace, el lote que reserva, la fecha y hora exacta del proceso.

**Figura 10:**Resultado del trigger



**Fuente:** Elaboración Propia

### 3.4.9 CONSULTAS COMPLEJAS

**Figura 11**:Muestra los trabajadores y cuando han vendido por mes

select  tbj.Id\_Trabajador,concat(per.Primer\_Nombre,' ',per.Segundo\_Nombre,' ',per.Apellido\_Paterno,' ',per.Apellido\_Materno) as nombre\_completo,

count(pln.Tipo\_Plan\_Pag) as Cantidad\_vendidos ,pln.Tipo\_Plan\_Pag,

case (month(ltv.Fecha\_Venta))

when 1 then 'Enero'

when 2 then 'Febrero'

when 3 then 'Marzo'

when 4 then 'Abril'

when 5 then 'Mayo'

WHEN 6 THEN 'Junio'

when 7 then 'Julio'

when 8 then 'Agosto'

When 9 then 'Septiembre'

when 10 then 'Octubre'

when 11 then 'Noviembre'

else 'Diciembre'

END

as mes,pln.Gestion\_Deuda

from plan\_de\_pago pln

inner join lote\_vendido ltv on pln.Id\_Lote\_Vendido = ltv.Id\_Lote\_Vendido

inner join trabajador tbj on pln.Id\_Trabajador = tbj.Id\_Trabajador

inner join persona per on per.Ci\_Identidad = tbj.Ci\_Identidad

group by pln.Tipo\_Plan\_Pag,nombre\_completo,pln.Gestion\_Deuda

Consulta que muestra el reporte de Trabajadores para saber quién vendió más lotes por mes.

**Fuente:** Elaboración Propia

### 3.4.10 PROCEDIMIENTOS ALMACENADOS

**Figura 12**:Procedimiento de almacenada(Muestra los lotes disponibles)

|  |
| --- |
| CREATE DEFINER=`root`@`localhost` PROCEDURE `Ver\_Lotes\_Disponibles` () SQL SECURITY INVOKER  SELECT lts.Id\_Lotes,urb.Ciudad,urb.Nombre\_Urbanizacion,lts.TamanhoMts2,lts.Precio,lts.Tipo\_Moneda,  case lts.Id\_EstadoLote  when 1 then 'Disponible'  when 2 then 'Reservado'  when 3 then 'vendido'  END as estado  from lotes lts inner join urbanizacion urb on lts.Id\_Urbanizacion = urb.Id\_Urbanizacion where lts.Id\_EstadoLote = 1$$  DELIMITER ; |

**Fuente:** Elaboración Propia

## 3.5 MODELO DE IMPLEMENTACIÓN

### 3.5.1. MODELO DE COMPONENTES

#### 3.5.1.1. CASO “REGISTRAR CLIENTES POTENCIALES”

**Figura 13:** Modelo de componente sobre el caso de uso de Registrar Clientes potenciales



**Fuente:** Elaboración Propia

#### 3.5.1.2. CASO “DETALLE MATERIAL”

**Figura 14:** Modelo de componente sobre caso de Registrar Reserva



**Fuente:** Elaboración Propia

### 3.5.2. MODELO DE DESPLIEGUE

#### 3.5.2.1. CASO “Registrar Clientes potenciales”

**Figura 15:** Modelo de despliegue sobre el caso de Registrar Clientes potenciales



**Fuente:** Elaboración Propia

#### 3.5.2.2. CASO “Registrar Reserva”

**Figura 16:** Modelo de despliegue sobre el caso de Registrar Reserva



**Fuente:** Elaboración Propia

**Figura 17:** Diseño de arquitectura física

**deployment Componente**

**Equipo de trabajo**

**Navegador web**

**Google Chrome**

**Firefox**

**Internet**

**Proveedor de**

**servicios**

**Cliente Web**

**Navegador Web**

**Server**

**templates**

**Logica**

**Base de Datos**

**MySql**

TCP/IP

HTTP/HTTPS

HTTP

HTTP/HTTPS

**Fuente:** Elaboración propia

**3.5.2Modelo de pruebas**

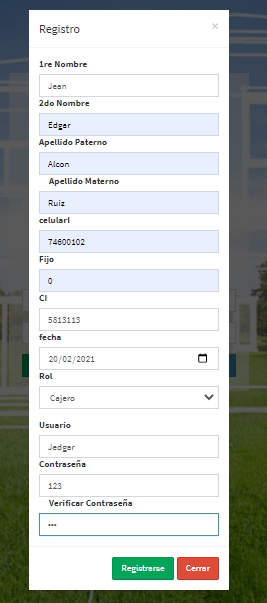
### 3.5.3. IMPLEMENTACIÓN DE UN REQUERIMIENTO

**Figura 18:** Diseño de login



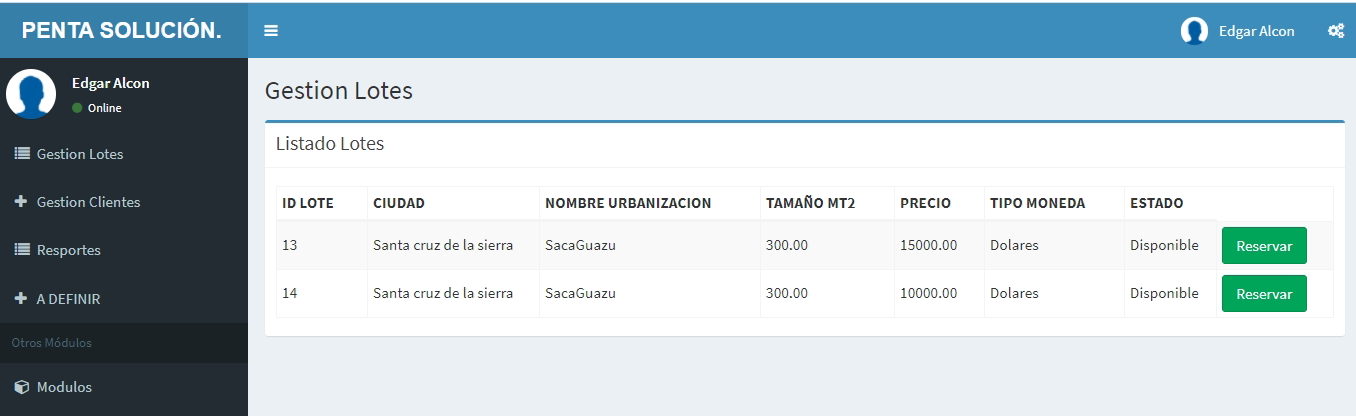
**Fuente:** Elaboración propia

**Figura 19:** Diseño de Registro para Trabajadores, usuario y contraseña



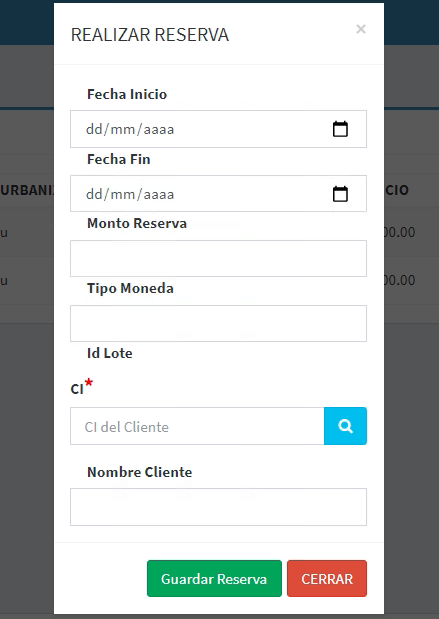
**Fuente:** Elaboración propia

**Figura 20:** Menú del Administrador



**Fuente:** Elaboración propia

**Figura 21:** Realizar Reserva



**Fuente:** Elaboración propia

**Figura 22:** Registrar Cliente Potencial o Cliente



**Fuente:** Elaboración propia

### 3.5.4. PLAN DE BACKUP DE LA BASE DE DATOS

Los Backup se crean principalmente como una medida de contingencia, o también para mover o copiar un proyecto en etapa de desarrollo.

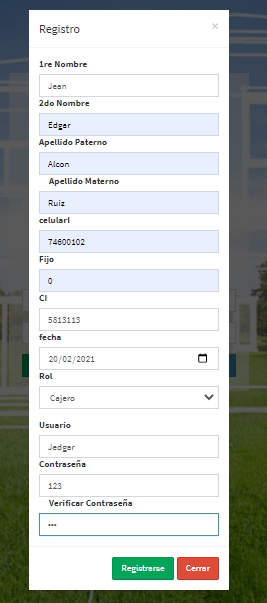
Se realizarán el siguiente tipo de Backup:

* **Backup Completo**: Se crea el backup completo (de toda la Base de datos el día sábado a las 11:30 pm.).
* **Backup Diferencial:** Se almacena la parte que ha cambiado con respecto al último Backup completo y se llevará a cabo de lunes a viernes a las 11:59pm.

## 3.5 MODELO DE PRUEBAS

### 3.5.1 PRUEBA FUNCIONAL

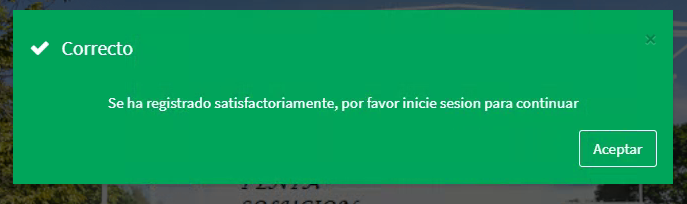
**Figura 23:** Prueba funcional RegistroTrabajador Para login



**Fuente:** Elaboración propia

Se ingresan los datos, para poder registrar a un trabajador para que pueda tener acceso al sistema.

**Figura 24:** Prueba funcional RegistroTrabajador Para login (Guardado exitoso)

**Fuente:** Elaboración propia

Mensaje de operación exitosa, luego de esto el usuario tendrá acceso al sistema

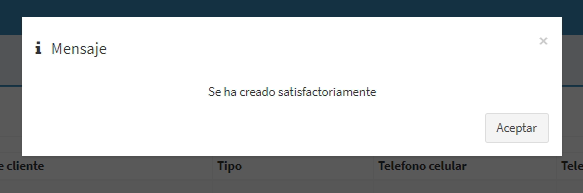
**Figura 25:** Prueba funcional Registro Cliente



**Fuente:** Elaboración propia

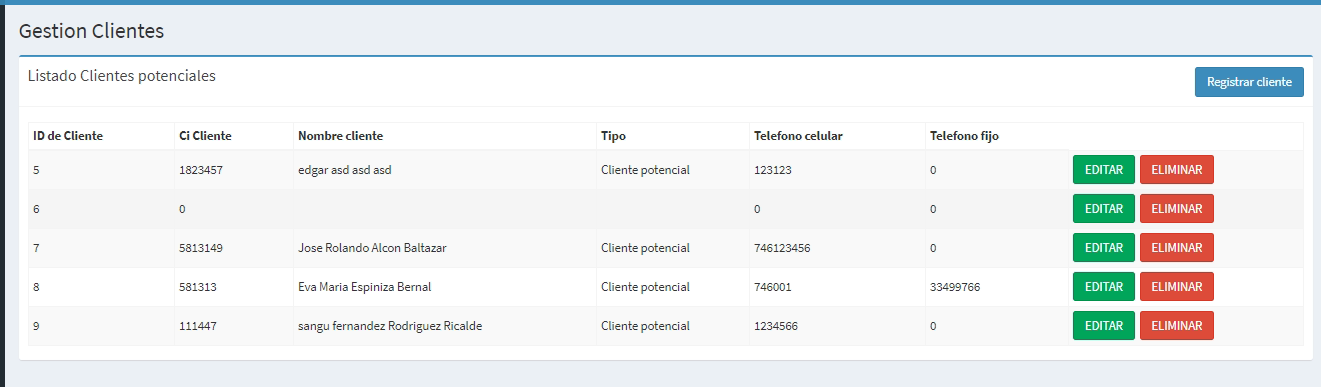
FrmRegistrar Clientes permite insertar nuevos clientes a sistema para tenerlos como potenciales clientes.

**Figura 26:** Prueba funcional Registro Cliente(Registro exitoso)

**Fuente:** Elaboración propia

Una vez el registro es exitoso, el sistema muestra diciendo el éxito de la operación.

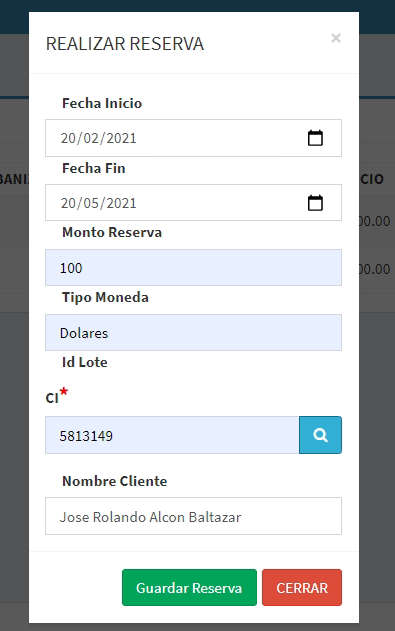
**Figura 27:** Prueba funcional Registro Cliente



**Fuente:** Elaboración propia

Muestra una lista de clientes Registrados, luego de todo el proceso.

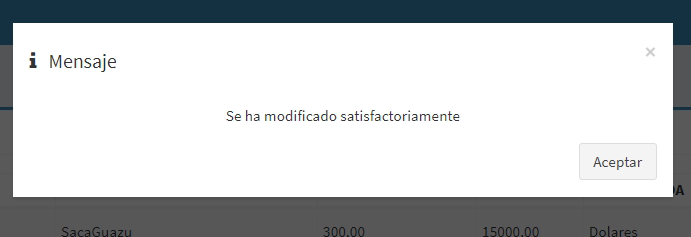
**Figura 28:** Prueba funcional Realizar Reserva



**Fuente:** Elaboración propia

Rellenar los datos para la reserva que el programa pide, una vez todo los datos pues, dar click en guardar reserva.

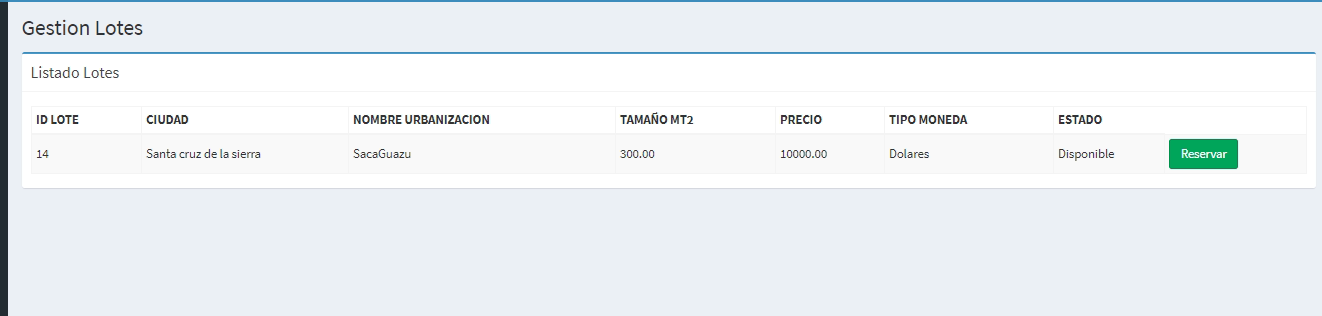
**Figura 29:** Prueba funcional Realizar Reserva



**Fuente:** Elaboración propia

Muestra un mensaje de Modificación satisfactorio por que se actualiza la tabla de lotes el estado se cambia

**Figura 30:** Prueba funcional Realizar Reserva



**Fuente:** Elaboración propia

Se muestra los lotes disponibles.

# CAPITULO IV

# 4 CONCLUSIONES Y RECOMENDACIONES

## 4.1 CONCLUSIONES

Se logró conseguir las metas del proyecto que son

* Se identificó y analizo los procesos de la institución “PENTA SOLUCION”.
* Se identificó todos los requisitos funcionales en base a los procesos administrativos
* Se realizó el modelo de requisitos, las capturas de los requisitos funcionales y no funcionales.
* Se diseñó el modelo de datos para la implementación de la base de datos.
* Se desarrolló la codificación de los puntos del sistema en base a los requisitos funcionales.
* Se evaluó las pruebas de funcionamiento del sistema codificado.

## 4.2 RECOMENDACIONES

**Recomendaciones generales**

Finalmente se ponen en consideración las siguientes recomendaciones con el propósito de que sirvan como punto de partida para proyectos de desarrollo de software:

* Implementar cifrado de datos para el inicio de sesión
* Proteger la integridad del código del sistema.
* Utilizar certificados SSL (HTTPS para la confianza de tus usuarios).

**Recomendaciones de usuario**

Se recomienda capacitar a personal capacitar para el uso del software.

# BIBLIOGRAFÍA

Aranguren, G. (10 de 12 de 2019). *Tópicos generales de ingeniería de sofware*.

Castillo, D. (8 de Octubre de 2013). *blog.Daniel Castillo*. Obtenido de http://jeanmarcodanielcastilloramos.blogspot.com/2013/10/metodologia-xp.html

eclases. (12 de 12 de 2019). *eclases*. Obtenido de eclases: http://eclases.tripod.com/id13.html

engine, S. t. (10 de 12 de 2019). *Smarty template engine*. Obtenido de Smarty template engine: https://www.smarty.net/docsv2/es/what.is.smarty.tpl

EPA. (09 de 09 de 2018). *EPA*. Obtenido de EPA: https://www3.epa.gov/ttncatc1/cica/help/hmonvals\_s.html

Francia, J. (17 de Agosto de 2017). *Scrum.org*. Obtenido de https://www.scrum.org/resources/blog/scrum-no-es-una-metodologia-es-un-framework

I., J., G., J. B., & Rumbaugh. (2000). *El Proceso Unificado de Desarrollo de Software.* Madrid: Pearson Educacion S.A.

iso25000. (12 de 12 de 2019). *ISO25000*. Obtenido de ISO25000: https://iso25000.com/index.php/normas-iso-25000

Joskowicz, J. (2008). *Reglas y Prácticas en eXtreme Programming .* España.

Martin, M. P. (10 de Diciembre de 2012). *LanceTalent*. Obtenido de https://www.lancetalent.com/blog/6-buenos-motivos-para-trabajar-con-php/

Menzinsky, A., López, G., & Palacio, J. (2019). *Scrum Master .* lubaris.

Raffino, M. E. (27 de Noviembre de 2018). *concepto.de*. Obtenido de https://concepto.de/lenguaje-de-programacion/

Romeu, I. A. (9 de Octubre de 2014). *El Blog del informatico*. Obtenido de https://albertoromeu.com/scrum-planning-poker/

Sutherland, K. S. (2013). *La Guía Definitiva de Scrum: las reglas de juego .*

Vila Grau, J. L. (14 de Agosto de 2019). *EL SIGNIFICADO DEL MANIFIESTO ÁGIL*. Obtenido de management plaza: https://managementplaza.es/blog/el-manifiesto-agil/

Wells, D. (2009). *http://www.extremeprogramming.org/values.html*.

# ANEXOS

**SCRIPT DE LA BASE DE DATOS**

CREATE TABLE `cliente` (

  `Id\_Cliente` int(11) NOT NULL,

  `Fecha\_Registro` date NOT NULL,

  `Comentario` varchar(100) DEFAULT NULL,

  `Ci\_Identidad` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `cuota` (

  `Id\_Cuota` int(11) NOT NULL,

  `Fecha\_Cuota` date NOT NULL,

  `Id\_Estadocuota` int(11) NOT NULL,

  `Monto\_Cuota` double(10,2) NOT NULL,

  `Monto\_Restante` double(10,2) NOT NULL,

  `Monto\_Total\_Pagar` double(10,2) NOT NULL,

  `Id\_Cliente` int(11) NOT NULL,

  `Id\_Plan\_Pago` int(11) NOT NULL,

  `Id\_Trabajador` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `estadocuota` (

  `Id\_Estadocuota` int(11) NOT NULL,

  `Nombre` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `estadolote` (

  `Id\_EstadoLote` int(11) NOT NULL,

  `Nombre` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `estadoreserva` (

  `Id\_EstadoReserva` int(11) NOT NULL,

  `Nombre` varchar(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `lotes` (

  `Id\_Lotes` int(11) NOT NULL,

  `Id\_EstadoLote` int(11) NOT NULL,

  `Latitud1` varchar(200) NOT NULL,

  `Latitud2` varchar(200) NOT NULL,

  `Latitud3` varchar(200) NOT NULL,

  `Latitud4` varchar(200) NOT NULL,

  `Longitud1` varchar(200) NOT NULL,

  `Longitud2` varchar(200) NOT NULL,

  `Longitud3` varchar(200) NOT NULL,

  `Longitud4` varchar(200) NOT NULL,

  `Numero\_Lote` int(11) NOT NULL,

  `Precio` double(10,2) NOT NULL,

  `TamanhoMts2` double(10,2) NOT NULL,

  `Tipo\_Moneda` varchar(20) NOT NULL,

  `Id\_Urbanizacion` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `lote\_vendido` (

  `Id\_Lote\_Vendido` int(11) NOT NULL,

  `Fecha\_Venta` date NOT NULL,

  `Id\_Cliente` int(11) NOT NULL,

  `Id\_Lotes` int(11) NOT NULL,

  `Id\_Trabajador` int(11) NOT NULL,

  `Id\_Estado\_LoteVendido` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `persona` (

  `Ci\_Identidad` int(11) NOT NULL,

  `Primer\_Nombre` varchar(50) NOT NULL,

  `Segundo\_Nombre` varchar(50) NOT NULL,

  `Apellido\_Paterno` varchar(50) NOT NULL,

  `Apellido\_Materno` varchar(50) NOT NULL,

  `Telefono\_Celular` int(11) NOT NULL,

  `Telefono\_Fijo` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `persona\_juridica` (

  `Id\_Persona\_Juridica` int(11) NOT NULL,

  `Fecha\_Registro` date NOT NULL,

  `Comentario` varchar(100) DEFAULT NULL,

  `Ci\_Identidad` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `plan\_de\_pago` (

  `Id\_Plan\_Pago` int(11) NOT NULL,

  `Fecha\_Inicio` date NOT NULL,

  `Fecha\_Fin` date NOT NULL,

  `Gestion\_Deuda` int(11) NOT NULL,

  `Monto\_Total` double(10,2) NOT NULL,

  `Tipo\_Moneda` varchar(20) NOT NULL,

  `Tipo\_Plan\_Pag` int(11) NOT NULL,

  `Id\_Lote\_Vendido` int(11) NOT NULL,

  `Id\_Cliente` int(11) NOT NULL,

  `Estado` int(11) NOT NULL,

  `Plazo` int(11) NOT NULL,

  `Id\_Trabajador` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `reserva` (

  `Id\_Reserva` int(11) NOT NULL,

  `Estado` int(11) NOT NULL,

  `Fecha\_Inicio` date NOT NULL,

  `Fecha\_Fin` date NOT NULL,

  `Monto\_Reserva` double(10,2) NOT NULL,

  `Tipo\_Moneda` varchar(20) NOT NULL,

  `Id\_Lotes` int(11) NOT NULL,

  `Id\_Trabajador` int(11) NOT NULL,

  `Id\_Cliente` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `roles` (

  `Id\_Role` int(11) NOT NULL,

  `Nombre\_Role` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `tipo\_pago` (

  `Id\_Tipo\_Pago` int(11) NOT NULL,

  `Nombre` varchar(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `trabajador` (

  `Id\_Trabajador` int(11) NOT NULL,

  `Fecha\_Inicio` date NOT NULL,

  `Usuario` varchar(100) NOT NULL,

  `Contraseha` varchar(100) NOT NULL,

  `Ci\_Identidad` int(11) NOT NULL,

  `Id\_Role` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `urbanizacion` (

  `Id\_Urbanizacion` int(11) NOT NULL,

  `Ciudad` varchar(50) NOT NULL,

  `ExtensionMts2` double(10,2) NOT NULL,

  `Nombre\_Urbanizacion` varchar(100) DEFAULT NULL,

  `Id\_Persona\_Juridica` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE EstadoPlanDePago(

Id\_Estado\_Plan\_Pago int not null primary KEY AUTO\_INCREMENT,

Nombre\_estado varchar(20)

);

CREATE TABLE Contrato(

Id\_Contrato int not null primary key AUTO\_INCREMENT,

Id\_Estado\_Contrato int not null,

Id\_Plan\_Pago int not null,

Id\_Cliente int not null,

Fecha\_Contrato date,

FOREIGN KEY (Id\_Estado\_Contrato) REFERENCES tipoContrato(Id\_Estado\_Contrato),

FOREIGN KEY (Id\_Cliente) REFERENCES cliente(Id\_Cliente),

FOREIGN KEY (Id\_Plan\_Pago) REFERENCESplan\_de\_pago(Id\_Plan\_Pago)

)