

UNIVERSIDAD ADVENTISTA DE BOLIVIA

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN WEB PARA EL PROCESO DE CÁLCULO
DE COSTOS Y CONTROL DE OBRAS CIVILES APLICANDO LA
ARQUITECTURA DE MICROSERVICIOS PARA LA EMPRESA J.B.B.L.

PROYECTO DE GRADO

Presentado como requisito para obtener

el grado académico de Licenciado

en Ingeniería de Sistemas

por

Ronald Luna Ramos

Tutor

Ing. Jhony Calle Cruz

Cochabamba, Julio de 2019

DEDICATORIA

Dedicado a mis padres Hernán Luna y Elvira Ramos que siempre estuvieron al pendiente y velando por mis estudios, de igual manera a mi tutor que me guio en el proyecto.

RESUMEN

Los procesos que tiene la empresa J.B.B.L. son manuales y usan un software basado en la arquitectura monolítica, que le origina varios tipos de problemas, por lo que decidieron implementar un nuevo sistema.

La finalidad del proyecto es desarrollar un sistema web basado en la arquitectura de microservicios desacoplando en pequeños subsistemas que trabajan de manera independiente y a su vez componen de un software totalmente funcional obteniendo así una mejor escalabilidad y mantenibilidad.

Para la recopilación de datos de la empresa se hizo varias entrevistas al dueño, obteniendo los requerimientos necesarios para el desarrollo del sistema.

Para el desarrollo del proyecto se utilizó la metodología Incremental porque permite trabajar por iteraciones y en cada una de ellas se hizo la entrega de partes del sistema dividido en subsistemas.

Los subsistemas se comunican entre sí intercambiando datos de forma segura y así obteniendo los resultados esperados.

Se hizo el uso del framework PHP de Laravel y como base de datos MariaDB, se utilizó los servicios de Web RESTful para el intercambio de datos y el estándar de JWT para la autenticación y comunicación entre subsistemas.

Palabras Clave: Arquitectura - Cálculo - Microservicios.

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1.....	2
EL PROBLEMA	2
1.1 Antecedentes.....	2
1.1.1 Antecedentes tecnológicos.....	4
1.2 Problema.....	4
1.2.1 Situación problemática	4
1.2.2 Formulación del problema	5
1.3 Objetivos	5
1.3.1 Objetivo general	5
1.3.2 Objetivos específicos	5
1.4 Alcances	6
1.5 Limites.....	6
1.6 Justificación.....	6
1.7 Análisis de Factibilidad.....	6
1.7.1 Factibilidad técnica.....	6
1.7.2 Factibilidad económica	7
1.7.3 Factibilidad operacional.....	8
CAPÍTULO 2.....	9
MARCO TEÓRICO	9
2.1 Métodos y técnicas de recopilación de información.....	9
2.1.1 Entrevista	9
2.1.1.1 Entrevistas no estructuradas	9
2.1.1.2 Entrevista formal.....	10
2.1.1.3 Entrevista focalizada	10
2.1.2 Encuesta	10
2.1.2.1 Encuestas abiertas	10
2.1.2.2 Encuestas cerradas o restringidas	10
2.1.2.3 Encuestas descriptivas.....	10

2.1.2.4	Encuestas explicativas.....	11
2.1.2.5	Encuestas longitudinales	11
2.1.3	La observación	11
2.1.3.1	El sujeto.....	11
2.1.3.2	El objeto	11
2.1.3.3	Los medios	11
2.1.3.4	Los instrumentos.....	11
2.1.4	Selección de método de recopilación de información	12
2.2	Modelado de Negocio	12
2.2.1	Business Model	12
2.2.2	Diagramas de Flujo.....	12
2.2.3	Selección de Modelado de Negocio	13
2.3	Presupuestos de obra.....	13
2.3.1	Características	13
2.3.2	Elaboración	14
2.3.2.1	Listado de precios básicos.....	14
2.3.3	Costos.....	15
2.3.3.1	Costo de directo.	15
2.3.3.2	Materiales.	16
2.3.3.3	Costos indirectos.....	16
2.4	Ingeniería de software	16
2.4.1	Metodologías de desarrollo de software	17
2.4.1.1	Modelo Incremental.....	18
2.4.1.2	Modelo Iterativo	20
2.4.1.3	Modelo Prototipado	21
2.4.1.4	Selección de metodología.....	22
2.4.2	UML	24
2.4.2.1	Versiones	24
2.4.2.2	Diagramas.....	25
2.4.3	Selección de versión de UML	25
2.5	Lenguajes de programación.....	26

2.5.1	PHP	26
2.5.1.1	Soporte para bases de datos:.....	26
2.5.2	Python	26
2.5.3	C#.....	27
2.5.4	Selección de lenguaje de programación.....	27
2.6	Entorno de trabajo	30
2.6.1	Laravel	30
2.6.1.1	Manejo de los datos en Laravel	31
2.6.1.2	Modelos.....	31
2.6.1.3	Vistas.....	31
2.6.1.4	Blade	32
2.6.2	Symfony	33
2.6.3	YII2.....	33
2.6.3.1	Usabilidad.....	33
2.6.4	Selección de entorno de trabajo.....	34
2.7	Arquitectura	36
2.7.1	Arquitectura monolítica	36
2.7.1.1	Escalabilidad vertical	36
2.7.2	Arquitectura orientada a servicios	36
2.7.2.1	Escalabilidad horizontal	37
2.7.3	Microservicios	37
2.7.3.1	Autónomo	37
2.7.3.2	Prioritario.....	38
2.7.3.3	Tolerancia a fallos.....	39
2.7.3.4	Gestión de Despliegues	39
2.7.3.5	Facilidad de implementación.....	40
2.7.4	Selección de arquitectura	40
2.8	Servicios Web	41
2.8.1	REST.....	41
2.8.1.1	Características.....	42
2.8.2	SOAP	43

2.8.3 Selección de Servicio Web	44
2.9 Base de datos	45
2.9.1 Mariadb	45
2.9.2 SQL Server.....	45
2.9.3 Selección de base de datos.....	46
2.10 Seguridad.....	47
2.10.1 Json Web Token	47
2.10.1.1 Usabilidad.....	47
2.10.1.2 Estructura.....	48
2.10.1.3 Funcionamiento	48
2.10.1.4 Uso	49
2.10.1.5 Sesiones.....	49
2.10.2 Selección de tipo de seguridad	50
2.11 Calidad de software.....	51
2.11.1 Calidad	52
2.11.2 Calidad de software	52
2.11.3 Pruebas unitarias.....	52
2.11.4 Pruebas de integración	52
2.11.5 Pruebas funcionales	53
CAPÍTULO 3.....	55
MARCO PRÁCTICO.....	55
3.1 Diseño de modelo del negocio.....	55
3.1.1 Modelo del negocio actual	55
3.1.2 Modelo de negocio alternativo	57
3.1.3 Planificación del desarrollo del proyecto.....	60
3.1.4 Diseño del caso de uso general del sistema.	60
3.2 Primer Incremento: Modelar la arquitectura de microservicios y desarrollar el subsistema de roles y usuarios	62
3.2.1 Análisis de requerimientos del primer incremento.....	62
3.2.1.1 Identificación y descripción de actores	63

3.2.1.2	Identificación de casos de uso por actor	64
3.2.1.3	Casos de casos del primer incremento	65
3.2.1.4	Diagramas de colaboración	68
3.2.2	Diseño del primer incremento	71
3.2.2.1	Diagrama de clases	71
3.2.2.2	Diseño de la base de datos.....	72
3.2.2.3	Diccionario de datos	73
3.2.2.4	Diseño de la arquitectura del software	77
3.2.3	Codificación del primer incremento	78
3.2.3.1	Implementación y funcionalidades	80
3.2.4	Casos de prueba del primer incremento.....	83
3.3	Segundo Incremento: Desarrollar el subsistema de stock de materiales.....	85
3.3.1	Análisis de requerimientos del segundo incremento	85
3.3.1.1	Identificación y descripción de actores.....	86
3.3.1.2	Identificación de casos de uso por actor	86
3.3.1.3	Casos de uso del segundo incremento.....	88
3.3.1.4	Diagramas de colaboración de Stock.....	90
3.3.2	Diseño del segundo incremento	93
3.3.2.1	Diagrama de clases	93
3.3.2.2	Diseño de base de datos.....	94
3.3.2.3	Diccionario de datos	95
3.3.2.4	Diseño de arquitectura de software.....	99
3.3.3	Codificación segundo incremento	100
3.3.3.1	Implementación y funcionalidades	102
3.3.4	Casos de prueba segundo incremento.....	104
3.4	Tercer Incremento: Desarrollar el subsistema de cálculo para el sistema web.	107
3.4.1	Análisis de requerimientos del tercer incremento	107
3.4.1.1	Identificación y descripción de actores.....	107
3.4.1.2	Identificación de casos de uso por actor	108
3.4.1.3	Diagramas de caso de uso por stock	109
3.4.1.4	Diagramas de colaboración del subsistema de cálculo	111

3.4.2	Diseño de tercer incremento.....	114
3.4.2.1	Diagrama de clases	114
3.4.2.2	Diseño de base de datos.....	115
3.4.2.3	Diccionario de datos	116
3.4.2.4	Diseño de arquitectura de software.....	120
3.4.3	Codificación del tercer incremento.....	121
3.4.3.1	Implementación y funcionalidades	122
3.4.4	Casos del prueba tercer incremento.....	124
3.5	Cuarto Incremento: Desarrollar el subsistema de fases para el sistema web.	126
3.5.1	Análisis de requerimientos del cuarto incremento	126
3.5.1.1	Identificación y descripción de actores	126
3.5.1.2	Identificación de casos de uso por actor	127
3.5.1.3	Diagramas de caso del cuarto incremento.....	127
3.5.1.4	Diagramas de colaboración de Fase.....	128
3.5.2	Diseño de cuarto incremento.....	128
3.5.2.1	Diagrama de clases	128
3.5.2.2	Diseño de base de datos.....	130
3.5.2.3	Diccionario de datos	131
3.5.2.4	Diseño de arquitectura de software.....	132
3.5.3	Codificación del cuarto incremento.....	134
3.5.3.1	Implementación y funcionalidades	134
3.5.4	Casos de prueba del cuarto incremento	136
	CONCLUSIONES	137
	RECOMENDACIONES	138
	REFERENCIAS BIBLIOGRÁFICAS	139
	ANEXOS	

Índice de Figuras

Figura N° 2. 1	12
Figura N° 2. 2	18
Figura N° 2. 3	22
Figura N° 2. 4	38
Figura N° 2. 5	39
Figura N° 3. 1	56
Figura N° 3. 2	58
Figura N° 3. 3	61
Figura N° 3. 4	63
Figura N° 3. 5	64
Figura N° 3. 6	68
Figura N° 3. 7	69
Figura N° 3. 8	69
Figura N° 3. 9	70
Figura N° 3. 10	70
Figura N° 3. 11	71
Figura N° 3. 12	72
Figura N° 3. 13	77
Figura N° 3. 14	78
Figura N° 3. 15	79
Figura N° 3. 16	80
Figura N° 3. 17	81
Figura N° 3. 18	81
Figura N° 3. 19	82
Figura N° 3. 20	82
Figura N° 3. 21	83
Figura N° 3. 22	86
Figura N° 3. 23	87
Figura N° 3. 24	91
Figura N° 3. 25	91

Figura N° 3. 26	92
Figura N° 3. 27	92
Figura N° 3. 28	92
Figura N° 3. 29	93
Figura N° 3. 30	94
Figura N° 3. 31	99
Figura N° 3. 32	100
Figura N° 3. 33	101
Figura N° 3. 34	102
Figura N° 3. 35	102
Figura N° 3. 36	103
Figura N° 3. 37	103
Figura N° 3. 38	104
Figura N° 3. 39	107
Figura N° 3. 40	108
Figura N° 3. 41	111
Figura N° 3. 42	112
Figura N° 3. 43	112
Figura N° 3. 44	113
Figura N° 3. 45	113
Figura N° 3. 46	114
Figura N° 3. 47	115
Figura N° 3. 48	120
Figura N° 3. 49	121
Figura N° 3. 50	122
Figura N° 3. 51	122
Figura N° 3. 52	123
Figura N° 3. 53	123
Figura N° 3. 54	124
Figura N° 3. 55	126
Figura N° 3. 56	127

Figura N° 3. 57	128
Figura N° 3. 58	129
Figura N° 3. 59	130
Figura N° 3. 60	133
Figura N° 3. 61	134
Figura N° 3. 62	135
Figura N° 3. 63	135

Índice de Tablas

Tabla N° 1. 1	7
Tabla N° 2. 1	22
Tabla N° 2. 2	27
Tabla N° 2. 3	28
Tabla N° 2. 4	35
Tabla N° 2. 5	441
Tabla N° 2. 6	45
Tabla N° 2. 7	47
Tabla N° 2. 8	50
Tabla N° 2. 9	50
Tabla N° 3. 1	62
Tabla N° 3. 2	65
Tabla N° 3. 3	65
Tabla N° 3. 4	66
Tabla N° 3. 5	67
Tabla N° 3. 6	67
Tabla N° 3. 7	68
Tabla N° 3. 8	73
Tabla N° 3. 9	74
Tabla N° 3. 10	75
Tabla N° 3. 11	76
Tabla N° 3. 12	76
Tabla N° 3. 13	83
Tabla N° 3. 14	89
Tabla N° 3. 15	88
Tabla N° 3. 16	88
Tabla N° 3. 17	89
Tabla N° 3. 18	90
Tabla N° 3. 19	95
Tabla N° 3. 20	95

Tabla N° 3. 21	96
Tabla N° 3. 22	97
Tabla N° 3. 23	98
Tabla N° 3. 24	105
Tabla N° 3. 25	109
Tabla N° 3. 26	109
Tabla N° 3. 27	110
Tabla N° 3. 28	110
Tabla N° 3. 29	116
Tabla N° 3. 30	117
Tabla N° 3. 31	118
Tabla N° 3. 32	118
Tabla N° 3. 33	119
Tabla N° 3. 34	124
Tabla N° 3. 35	127
Tabla N° 3. 36	131
Tabla N° 3. 37	132
Tabla N° 3. 38	136

GLOSARIO

Arquitectura: Estructura del software que tiene impacto directo con la capacidad que tendrá.

Microservicios: Enfoque para desarrollar un software en pequeños servicios.

JSON: Formato ligero para el intercambio de datos

JWT: Estándar de autenticación web en formato JSON utilizado para transferir datos entre cliente y servidor.

Payload: Carga de datos en el cuerpo de un JWT.

ORM: Mapeador de base de datos.

REST: Transferencia de Estado Representacional.

Http: Protocolo de transferencia de hipertexto.

XtraDB: Motor de base de datos de MariaDB

TDD: Desarrollo guiado por pruebas.

Framework: Marco de trabajo y conjunto de librerías estandarizadas.

API: funciones que dan la capacidad de usar un servicio web.

INTRODUCCIÓN

Con el pasar del tiempo las aplicaciones monolíticas van creciendo en tamaño, en código, la mantenibilidad y escalabilidad se hace más complicado, no importa sea una aplicación web o de escritorio. En algún momento estas tienen un límite, es por eso por lo que se propone una arquitectura horizontal, con el fin de que los subsistemas sean “autónomos” y cada uno de ellos aseguren su disponibilidad, confidencialidad e integridad.

El proyecto fue dividido en tres capítulos que se mencionara a continuación:

En el primer capítulo se describe todo el trabajo realizado por la empresa. luego se identifica los problemas y se propone los objetivos que ayudaran a dar una solución.

El capítulo dos contiene todas las bases de información que son de gran importancia para el desarrollo del proyecto como ser: la metodología, lenguaje de programación, base de datos, arquitectura, tipo de seguridad a implementarse.

En el capítulo tres se muestra los modelados de proceso actual y alternativos. Se trabaja con la metodología, lenguaje de programación, framework, base de datos que fueron seleccionadas en el capítulo dos. En la metodología se ve todos los incrementos identificados e implementados en el proyecto.

CAPÍTULO 1

EL PROBLEMA

1.1 Antecedentes

Al existir una alta demanda de construcciones, alza de precios en terrenos y casas surge una mayor necesidad de empresas constructoras. De igual manera el GOBIERNO AUTÓNOMO MUNICIPAL DE LA CIUDAD DE EL ALTO realiza obras para la población. Esta entidad contrata distintas empresas para realizar sus obras. En ese sentido el señor José Luis Huanca hace la apertura de J.B.B.L. con el fin de cumplir la demanda existente. La empresa tiene su oficina ubicada en la zona de Rio Seco Industrial 10 de la ciudad de El Alto, haciendo la apertura de esta en fecha 01 de febrero del año 2011.

La empresa brinda los servicios de construcción y/o remodelación de obras civiles cumpliendo con normas establecidas para la construcción. Entre las obras civiles que realiza son: construcción de sedes sociales, parques, plazas, cordones de acera, adoquinado de calles, casas, edificios o condominios.

Para el comienzo de una obra es necesario saber cuánto será el costo total, gastos directos e indirectos además el tiempo en que la obra será realizada. Para ello se sigue ciertos procedimientos con el objetivo de realizar dicho trabajo.

Inicialmente se hará lectura de los planos que fueron diseñados por algún arquitecto, de igual manera si hay documentos de requerimientos se los debe leer detalladamente para conocer que materiales usara. La empresa tiene una extensa lista de materiales que fueron llenados a través del tiempo, en cada lista de proveedor se encuentra un listado de los materiales de construcción que este le suministra. Los precios de una gran cantidad de materiales son actualizados periódicamente en un intervalo aproximado de 120 días, pero no todos son datos actuales porque no se pudo averiguar el precio de estas.

Estos registros son realizados en archivos Microsoft Excel y almacenados en la computadora de la empresa (ver Anexo 1). Consiguiente a esto se procede a calcular el costo del proyecto que es la tarea tediosa y difícil, lo que conlleva a dividirlo en pequeños módulos. En estos módulos se contempla los materiales y gastos como ser, pagos a mano de obra, gastos la maquinaria que

este requerirá y el transporte de materiales. Todo este trabajo es hecho en base a fórmulas que son diseñadas por el arquitecto o personal responsable a cargo.

Para diseñar las fórmulas se reúnen los encargados de área y cada uno analizan los aspectos y detalles. Con ello se toma en cuenta cantidad de material, tipo, mano de obra, la unidad de medida también si requerirá reusar otra fórmula y luego se procede a estimar el tiempo en que este será realizado.

En las fórmulas se debe tomar en cuenta los gastos directos e indirectos que afectaran al proceso de presupuesto.

Para el proceso de cálculo tienen un sistema de escritorio que trabaja en base a dimensiones predefinidas.

Una vez obtenido el costo total del proyecto se procede a estimar el cronograma basado a fechas. Los cronogramas de trabajo son tiempos determinados en fecha de inicio y fecha final y son realizados en Microsoft Excel (Ver Anexo 2). Consiguiente a eso se prepara un informe indicando el costo total de la obra y el cronograma, establecido en fechas de entregas de avances indicando todos los aspectos detalladamente, luego se envía a la entidad contratante.

Si el proyecto presupuestado está dentro del alcance previsto se da el visto bueno y se realiza la firma de contrato hecho por la persona jurídica o natural contratante y la empresa J.B.B.L.

Una vez obtenido el proyecto se hace la compra de material necesario que luego son llevados al almacén que tiene la empresa.

Posterior a esto se hace el reclutamiento de personal entre los que comprenden son: albañiles maestros, ayudantes y arquitectos, supervisores de obra, personal de limpieza y cocineras. A estos se les hacen un contrato de trabajo por jornal¹. Posterior a eso se procede con el alquiler de maquinaria que será necesaria para el proyecto.

Para iniciar un proyecto se hace una reunión general con todo el personal involucrado en el proyecto, en los cuales se aborda temas de seguridad, relacionamiento laboral, principios y valores. De igual manera se hace conocer los objetivos establecidos por la empresa.

¹ Jornal: Cantidad de dinero que gana un trabajador por cada día de trabajo.

Se realiza el cronograma de entregas establecido a fechas, en base a este cronograma es que se va entregando el proyecto por partes.

Para hacer un seguimiento a las obras se designa un supervisor de obra. Este hace una revisión de la obra diariamente para verificar el estado de avance de cada módulo, lleva en mano un “Registro de obra” y hace el anote de todos los trabajos realizados, posterior a esto los trabajos son verificados con el plazo establecido en el contrato.

El cronograma muestra el proyecto dividido en módulos y cada una de ellas tiene una fecha de inicio y una fecha final.

En muchas ocasiones puede que estos registros de avance sean extraviados y en otras situaciones son dañados, dificultando el control de progreso de trabajo de cada módulo.

Una vez concluido el proyecto se hace la entrega invitando a todo el personal en un acto ofrecido por la empresa.

1.1.1 Antecedentes tecnológicos

- Actualmente en Bolivia existe un sistema de escritorio llamado “PRESOM” para el proceso de cálculos de presupuestos de obra con una licencia de Bs. 600 anual por computadora.
- Quark de igual manera es un software de escritorio que genera presupuestos de obra.
- Los softwares² mencionados anteriormente fueron realizados en una arquitectura monolítica.

1.2 Problema

En este apartado se desglosa los problemas encontrados.

1.2.1 Situación problemática

Los problemas identificados se mencionan a continuación

- La búsqueda manual de materiales en archivos Microsoft Excel para una obra provoca una lentitud en la obtención de lista de materiales a utilizarse.

² Software: Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora

- La información irreal en el costo de materiales que se utilizará en la obra provoca cálculos errados.
- Los procedimientos manuales de seguimiento a obras tienden a deteriorarse con el pasar del tiempo lo que provoca desorganización en el trabajo.

1.2.2 Formulación del problema

Los deficientes procedimientos manuales en el proceso de cálculo de costos y control de obras civiles en la empresa J.B.B.L. provoca pérdida de material de construcción y gasto adicional en pagos a maquinarias, herramientas e información incompleta.

1.3 Objetivos

En este apartado se define los objetivos a tratarse en el proyecto.

1.3.1 Objetivo general

Desarrollar un sistema de información web para el proceso de cálculo de costos y control de obras civiles aplicando la arquitectura de microservicios para la empresa JBBL.

1.3.2 Objetivos específicos

- Diseñar el modelo de negocio alternativo para el proceso de cálculo de presupuesto de obras de acuerdo con las fases de esta.
- Asegurar la integración de cada uno de los subsistemas y el intercambio de mensajes mediante el uso de token de autenticación.
- Proponer un sistema arquitectural de fácil mantenibilidad y gran escalabilidad horizontal, mediante la implementación de la arquitectura basada en microservicios.
- Desarrollar el subsistema de autenticación y usuarios.
- Desarrollar el subsistema de stock.
- Desarrollar el subsistema de cálculo de presupuestos de obras.
- Desarrollar el subsistema de seguimiento a obras basado en fechas establecidas de entregas.

1.4 Alcances

El sistema incluye los subsistemas de:

- Gestionar presupuestos múltiples de obras.
- Gestionar usuarios, subsistemas y roles.
- Gestionar productos por categoría y unidad.
- Gestionar proveedores de materiales para construcción a la empresa.
- Realizar cálculos exactos por obra y detallado por obra civil.
- Gestionar gastos que serán realizados por obra.
- Realizar un control de obra establecido en base a inicio y entrega.

1.5 Limites

No se agrega ningún subsistema de Recursos Humanos para empleados, de igual manera no tendrá el subsistema de control de activos fijos, ni el subsistema de ventas.

1.6 Justificación

El sistema será de gran utilidad para la empresa JBBL porque eliminará todo un proceso innecesario usado para el cálculo de presupuesto, ahorrará bastante tiempo y dinero a la empresa. La empresa tendrá información detallada y actualizada a la mano de materiales, proyectos, avances de obra y también podrá hacer seguimiento a ellos.

Se podrá obtener información detallada de cada proyecto registrado en el sistema.

1.7 Análisis de Factibilidad

A continuación, se describe la factibilidad del proyecto.

1.7.1 Factibilidad técnica

Para desarrollar un software es necesario conocer distintas herramientas, las cuales se listarán a continuación.

- MariaDB

- Framework Laravel 5
- Visual Code
- Postman
- JWT
- HTML, CSS, JAVASCRIPT, AJAX
- DbForge Studio

Los requerimientos de hardware para la utilización del sistema son:

- Procesador Intel Core i3.
- Memoria RAM 4GB.
- Sistema Operativo Windows 8 o superior de 64 Bits.

El postulante tiene las capacidades y conocimientos de las diferentes herramientas para elaborar el proyecto y la empresa tiene el ambiente y condiciones para implementar el sistema.

1.7.2 Factibilidad económica

A continuación, se detallará el costo del proyecto.

Tabla N° 1. 1

Precios de hosting y dominio

Descripción	Precio
Hosting Anual	Bs. 350
Dominio	Bs. 300
TOTAL	Bs. 650

Fuente: [Elaboración propia]

Al ser un proyecto de grado, el postulante desarrollador no percibirá monto de dinero.

En la tabla 1.1 se mencionó las herramientas que se usara con su correspondiente detalle con lo que se llega a la conclusión de que el sistema si es económicamente factible.

1.7.3 Factibilidad operacional

No es necesario tener un conocimiento avanzado en computación para hacer uso de la herramienta. El sistema es intuitivo además los conceptos responsivos serán aplicados al proyecto para ser desplegados en diferentes dispositivos y resoluciones.

El sistema está diseñado para funcionar exclusivamente en un entorno web ya sea en Google Chrome o Mozilla Firefox que esté instalado en una computadora o una Tablet, se hará una capacitación al usuario acerca del uso, por lo cual el sistema es factible operacionalmente.

CAPÍTULO 2

MARCO TEÓRICO

En el presente capítulo se desarrolla el fundamento teórico de métodos, técnicas de recolección de información, metodología de desarrollo de software, análisis y diseño, arquitectura, lenguajes de programación, base de datos y seguridad que serán utilizadas durante el desarrollo del proyecto.

2.1 Métodos y técnicas de recopilación de información

Con el fin de realizar una investigación es necesario una cercana interacción con el usuario final para levantar los requerimientos que este solicita, en lo cual se aplican distintas herramientas y métodos para la recolección de información entre las que se encuentran: observación, entrevistas y encuestas.

Por otra parte, Jesús Contreras afirma que cuando se lleva a cabo un trabajo de investigación es necesario considerar los métodos, las técnicas e instrumentos como aquellos elementos que aseguran el hecho empírico de la investigación. [1]

2.1.1 Entrevista

Las entrevistas son usadas para recolectar información mediante preguntas en los que actúan dos sujetos que son el entrevistador y el entrevistado.

Grajales indica que la entrevista es la relación personal entre dos o más sujetos en el cual el entrevistado posee información que interesa al entrevistador. Esta puede ser libre, en la que la elección de los temas sale espontáneamente, dirigida, donde el entrevistador hace una selección previa de los temas de interés para él y así dirige la conversación y la estandarizada en la que el entrevistador lee un formato del cual no puede salirse.[1]

2.1.1.1 Entrevistas no estructuradas

En una entrevista no estructurada existe una libertad para formular las preguntas y respuestas. No son guiadas por un cuestionario o modelo rígido.

2.1.1.2 Entrevista formal

Es la modalidad menos estructurada posible de entrevista, ya que se reduce a una simple conservación sobre el tema en estudio. Lo importante no es definir los límites de lo tratado ni ceñirse a algún esquema previo, sino "hacer hablar" al entrevistado, de modo de obtener un panorama de los problemas más salientes, de los mecanismos lógicos y mentales del respondiente, de los temas que para él resultan de importancia. [1]

2.1.1.3 Entrevista focalizada

Es prácticamente tan libre y espontánea como la anterior, pero tiene la particularidad de concentrarse en un único tema. El entrevistador deja hablar sin restricciones al entrevistado, proponiéndole apenas algunas orientaciones básicas, pero cuando éste se desvía del tema original, el entrevistador vuelve a centrar la conversación sobre el primer asunto. [1]

2.1.2 Encuesta

Con esta técnica de recolección de datos da lugar a establecer contacto con las unidades de observación por medio de los cuestionarios previamente establecidos. Esta es una modalidad utilizada por las empresas de mercadeo o institutos de opinión, en muchas ocasiones existe polémicas y controversias.

Existen distintos tipos de encuestas y los mencionamos a continuación.

2.1.2.1 Encuestas abiertas

Hugo Cerda indica que las encuestas abiertas o no restringidas propician respuestas que podemos calificar como espontáneas y libres. Suelen ser más profundas, más argumentadas y ricas, pero presentan la desventaja de que se limita mucho la tabulación de éstas. [2]

2.1.2.2 Encuestas cerradas o restringidas

Estas incitan a responder de forma prevé y especifica las respuestas formuladas por el encuestador.

2.1.2.3 Encuestas descriptivas

Estas son las más comunes, es útil para generar información de un grupo amplio de la población.

2.1.2.4 Encuestas explicativas

El autor Hugo Cerda indica que estas no difieren mayormente de las investigaciones explicativas, pero en el primer caso tienen una dimensión o un alcance masivo. Buscan explicar las causas de un fenómeno o saber por qué ocurren las cosas, cuáles son los factores determinantes, de dónde proceden, cómo se transforman, etc. [2]

2.1.2.5 Encuestas longitudinales

Cualquier estudio longitudinal, de lo cual no son ajenas las encuestas, se caracteriza porque estudia los fenómenos y los hechos en desarrollo, en el tiempo o en un determinado período de él, ya sea para describir o caracterizar sus aspectos importantes o para establecer sus factores asociados. [2]

2.1.3 La observación

Posiblemente esta podría ser uno de los métodos más utilizados y antiguos dentro de la investigación científica. Debido a que este es más fácil de aplicar [2]

La observación se traduce en un registro visual de lo que ocurre en el mundo real.

2.1.3.1 El sujeto

No es otra cosa que el observador, o sea la persona o las personas que observan los fenómenos o las cosas seleccionadas con tal propósito. [2]

2.1.3.2 El objeto

Este es el observado por el sujeto.

2.1.3.3 Los medios

Pueden ser diversos, ya que pueden ser a través de la vista, oído o cualquier elemento que permita percibir los fenómenos que ocurren entorno al objeto.

2.1.3.4 Los instrumentos

Son medios de apoyo que ayudan en la observación, pueden ser medios escritos, es decir toda tecnología que permita registrar los acontecimientos observados.

2.1.4 Selección de método de recopilación de información

Para el método de recopilación se decidió hacer uso de entrevista no formalizada. Descartando la observación, porque el objeto de estudio se encuentra en otra ciudad, también se descartó la encuesta por que las preguntas y respuestas son un tanto más cerradas que la entrevista y no entregan información suficiente para la elaboración del proyecto.

2.2 Modelado de Negocio

Mediante el modelado de negocio se define reglas, flujo de trabajo y para esto se usa diferentes diagramas para el modelo del negocio que se describirá a continuación.

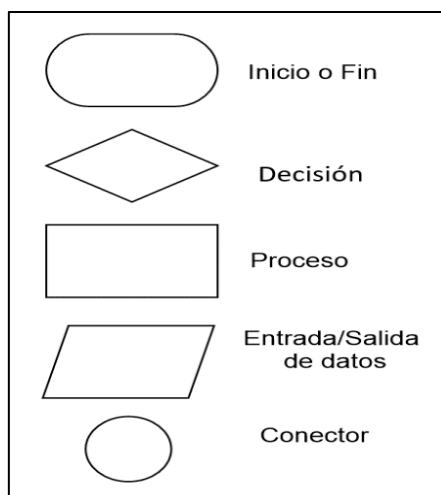
2.2.1 Business Model

Las organizaciones comprenden cada vez más que cumplir con sus ambiciones de sostenibilidad no solo requiere nuevas tecnologías, sino también innovación a nivel de modelo de negocio. Para facilitar el diseño de modelos de negocio más sostenibles, una gama de nuevas herramientas y se han desarrollado técnicas. Si bien esto resultó en el diseño de una amplia gama de modelos de negocios prometedores, solo muy pocos son implementados exitosamente [3].

2.2.2 Diagramas de Flujo

Los diagramas de flujo permiten ver de manera general un determinado proceso. En la siguiente figura se detalla los componentes de un diagrama de flujo.

Figura N° 2. 1



Fuente: [Elaboración Propia]

Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de la operación, en pocas palabras son la representación simbólica de los procedimientos administrativos. [4]

En la siguiente imagen vemos las componentes de un diagrama de flujo detallando cada paso que este debe seguir.

2.2.3 Selección de Modelado de Negocio

Tabla 2. 1

Comparativa de modelos de negocio

Modelo	Características	Ventajas	Desventajas
Business Model	Conoce y optimiza las actividades.	Un enfoque más gráfico.	Muy poca aceptación.
Diagramas de Flujo	Remplazo de texto.	Sencillo de utilizar. Fácil de comprender.	Cuellos de botella.

Fuente: [Elaboración Propia]

Se eligió el uso de diagramas de flujo por ser sencillo de usar y fácil implementación porque este nos permite entender rápidamente el flujo de trabajo de la entidad, y las operaciones que este tendrá.

2.3 Presupuestos de obra

Presupuestar una obra, es establecer de qué está preparada y cuántas unidades de cada componente se solicitan para aplicar precios a cada uno de estos. Luego se obtiene su valor en el instante que se requiera.

2.3.1 Características

Todo presupuesto tiene cuatro características primordiales: es aproximado, es singular, es temporal y es una herramienta de control.

El presupuesto es aproximado, sus previsiones se acercarán más o menos al costo real de la obra, dependiendo de la habilidad (uso correcto de técnicas presupuestales), el criterio (visualización correcta del desarrollo de la obra) y experiencia del presupuestador.

El presupuesto es singular, como lo es cada obra, las condiciones de localización, clima y medio ambiente, calidad de la mano de obra características del constructor, etc. Cada obra requiere un presupuesto propio, así como cada persona o empresa tiene su forma particular de presupuestar. [5]

El presupuesto es temporal, los costos que en él se establecen sólo son válidos mientras tengan vigencia los precios que sirvieron de base para su elaboración. Los principales factores de variación son: Incremento del costo de los insumos y servicios; utilización de nuevos productos y técnicas; desarrollo de nuevos equipos, herramientas, materiales, tecnología, etc.

El presupuesto es un instrumento de control, permite correlacionar la realización presupuestal con el avance físico, su comparación con el costo real permite detectar y corregir fallas y prevenir causales de variación por ajuste en alcances o cambios en actividades. No debe concebirse como un documento estático, cuya función concluye una vez elaborado. El presupuesto de construcción se debe estructurar como un instrumento dinámico, que además de confiable y preciso sea fácilmente controlable para permitir su actualización sistemática y evitar que se convierta en una herramienta obsoleta y de poca utilidad práctica [5].

2.3.2 Elaboración

Se realiza en base a los planos y en las especificaciones técnicas de un proyecto, además de otras condiciones de ejecución, se elaboran los cálculos de los trabajos a ejecutar, se hacen los análisis de precios unitarios de los diversos ítems y se establecen los valores parciales de los capítulos en que se agrupan los ítems, y así obtener el valor total de la obra. Los pasos para seguir son:

2.3.2.1 Listado de precios básicos

El presupuesto debe incluir la lista de precios básicos de materiales, equipos y salarios utilizados.

- **Análisis unitarios:** Incluye indicaciones de cantidades y costos de materiales, transportes, desperdicios, rendimientos, costo de mano de obra, etc.
- **Presupuesto por capítulos:** Los costos de obra se presentan divididos por capítulos de acuerdo con el sistema de construcción, contratación, programación, etc.
- **Componentes del presupuesto:** Se presenta el desglose del presupuesto con las cantidades y precios totales de sus componentes divididos así: materiales, mano de obra, subcontratos, equipos y gastos generales. Finalmente, en: costos directos y costos indirectos.
- **Fecha de presupuesto:** Se debe indicar la fecha en la que se hace el estimativo, en caso de haber proyecciones de costos en el tiempo, se deben indicar.[4]

2.3.3 Costos

En general se pueden identificar los siguientes grandes componentes los cuales participan en los costos básicos de una obra: Materiales, mano de obra, equipos y herramientas, gastos generales: administración e imprevistos.

Los gastos generales también se conocen como costos indirectos, están relacionados especialmente con el tiempo de ejecución, e incluyen todos aquellos factores diferentes de los costos directos, que afectan la ejecución de la obra incluyendo gastos administrativos, de mantenimiento, financieros, impuestos, pólizas, servicios públicos, comunicaciones, control técnico, campamentos, vías de acceso, etc., además de los imprevistos. [5]

2.3.3.1 Costo de directo.

El costo directo del precio unitario de cada ítem debe incluir todos los costos en que se incurre para realizar cada actividad, en general, este costo directo está conformado por tres componentes que dependen del tipo de ítem o actividad que se esté presupuestando. (excavación, hormigón armado para vigas, replanteo, etc.). [5]

- **Materiales:** Es el costo de los materiales puestos en obra.
- **Mano de Obra:** Es el costo de la mano de obra involucrada en el ítem, separado por cada especialidad, por ejemplo, en el caso de una viga de hormigón armado se necesita la participación de albañil, encofrador.
- **Maquinaria, equipo y herramientas:** Es el costo de los equipos, maquinarias y todas las

herramientas utilizadas en el ítem que se está analizando.

Seguidamente se presenta la metodología para determinar los costos de cada uno de los componentes del costo directo.

2.3.3.2 Materiales.

Los materiales son los recursos que se utilizan en cada una de las actividades o ítems de la obra. Están determinados por las especificaciones técnicas, donde se define la calidad, cantidad, marca, procedencia, color, forma, o cualquier otra característica necesaria para su identificación.

- **Costo de los Materiales:** El costo de los materiales consiste en una cotización adecuada de los materiales a utilizar en una determinada actividad o ítem, esta cotización debe ser diferenciada por el tipo de material y buscando al proveedor más conveniente. El precio para considerarse debe ser el puesto en obra, por lo tanto, este proceso puede ser afectado por varios factores tales como: costo de transporte, formas de pago, volúmenes de compra, ofertas del momento, etc. [5]

2.3.3.3 Costos indirectos.

Los costos indirectos son aquellos gastos que no son fácilmente cuantificables como para ser cobrados directamente al cliente.

Los costos indirectos incluyen: gastos generales, utilidades y los impuestos.

- **Gastos Generales:** Son aquellos gastos no incluidos en los costos directos y son muy variables, dependiendo de aspectos como el lugar donde se debe realizar la obra. Así, por ejemplo, las obras locales tienen gastos generales más bajos que los que están ubicados en el campo y también es obvio que una empresa constructora grande tiene gastos generales mayores que la de una pequeña.

2.4 Ingeniería de software

La ingeniería de software es un conjunto de herramientas y métodos para el desarrollo de sistemas informáticos.

El autor Presman dice que la ingeniería de software es una disciplina o área de la informática o ciencia de la computación, que ofrece técnicas y métodos para desarrollar y mantener software de calidad que resuelva todo tipo. [6]

Por otra parte, Jacobson indica que la ingeniería de software se define como un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo, en este caso, la obtención de un producto de software de calidad [7]

Con el transcurso de los años se han desarrollado recursos que conforman la ingeniería del software, es decir, herramientas y técnicas de especificación, diseño e implementación del software: la programación estructurada, la programación orientada a objetos, las herramientas CASE, la documentación, los estándares, los servicios web, el lenguaje UML, etc.

Un método de ingeniería de software es un enfoque estructurado para desarrollar software cuyo objetivo es facilitar la producción de productos software de alta calidad a un coste razonable. Indican cómo construir técnicamente el software. Los métodos abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requerimientos del sistema y del software, diseño de procedimientos algorítmicos, codificación, prueba y mantenimiento. [5]

2.4.1 Metodologías de desarrollo de software

Una metodología de desarrollo de software se refiere a un framework (entorno o marco de trabajo) que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información. [8]

A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad.

El framework para metodología de desarrollo de software consiste en:

- Una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software.
- Herramientas, modelos y métodos para asistir al proceso de desarrollo de software.

Estos framework son a menudo vinculados a algún tipo de organización, que además desarrolla, apoya el uso y promueve la metodología. [8]

Cada metodología tiene un enfoque propio al desarrollo de software los cuales desarrollados en metodologías que son utilizadas para la planeación y desarrollo del software.

2.4.1.1 Modelo Incremental

El modelo incremental fue propuesto por Harlan Mills en el año 1980. Surgió el enfoque incremental de desarrollo como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema. [9]

Este modelo se conoce también bajo las siguientes denominaciones:

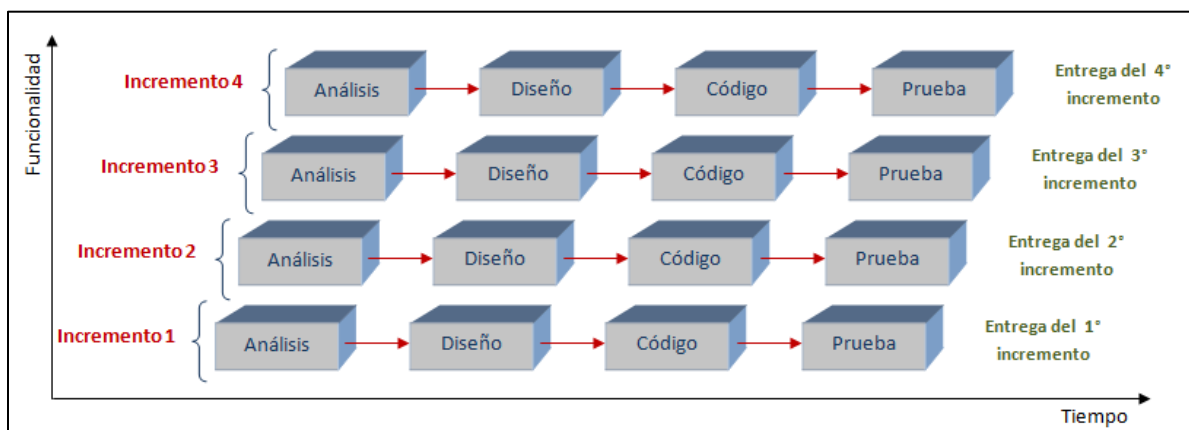
- Método de las comparaciones limitadas sucesivas.
- Ciencia de salir del paso.
- Método de atacar el problema por ramas.

El Modelo Incremental combina elementos del Modelo Lineal Secuencial con la filosofía interactiva de Construcción de Prototipos. Como se muestra en la Figura 2.2, el modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software. El primer incremento generalmente es un producto esencial denominado núcleo.

En una visión genérica, el proceso se divide en 4 partes: análisis, diseño, código y prueba

Figura N° 2. 2

Modelo Incremental



Fuente: [9, Fig. 1]

Sin embargo, para la producción del Software, se usa el principio de trabajo en cadena o Pipeline. Con esto se mantiene al cliente en constante contacto con los resultados obtenidos en cada incremento. Es el mismo cliente el que incluye o desecha elementos al final de cada

incremento a fin de que el software se adapte mejor a sus necesidades reales. El proceso se repite hasta que se elabora el producto completo. De esta forma el tiempo de entrega se reduce considerablemente.

El Modelo Incremental es de naturaleza interactiva brindando al final de cada incremento la entrega de un producto completamente operacional. Este modelo es particularmente útil cuando no se cuenta con una dotación de personal suficiente. Los primeros pasos los pueden realizar un grupo reducido de personas y en cada incremento se añadirá personal, de ser necesario. Por otro lado, los incrementos se pueden planear para gestionar riesgos técnicos.

Durante el proceso se trata de llevar a cabo al proyecto en diferentes partes que al final terminará siendo la solución completa requerida por el cliente, pero éstas partes no se pueden realizar en cualquier orden, sino que dependen de lo que el cliente este necesitando con más urgencia, de los puntos más importantes del proyecto, los requerimientos más básicos, difíciles y con mayor grado de riesgo, ya que estos se deben hacer al comienzo, de manera que se disminuya la dificultad y el riesgo en cada versión. [9]

De este modo podemos terminar una aplicación ejecutable (primera versión) que podrá ser entregada al cliente para que éste pueda trabajar en ella y el programador pueda considerar las recomendaciones que el cliente efectúe para hacer mejoras en el producto. Estas nuevas mejoras deberán esperar a ser integradas en la siguiente versión junto con los demás requerimientos que no fueron tomados en cuenta en la versión anterior.

El modelo incremental consiste en un desarrollo inicial de la arquitectura completa del sistema, seguido de sucesivos incrementos funcionales. Cada incremento tiene su propio ciclo de vida y se basa en el anterior, sin cambiar su funcionalidad ni sus interfaces. Una vez entregado un incremento, no se realizan cambios sobre el mismo, sino únicamente corrección de errores. Dado que la arquitectura completa se desarrolla en la etapa inicial, es necesario conocer los requerimientos completos al comienzo del desarrollo.

Al iniciar del desarrollo, los clientes o los usuarios, identifican a grandes rasgos, las funcionalidades que proporcionará el sistema. Se confecciona un bosquejo de requisitos funcionales y será el cliente quien se encarga de priorizar que funcionalidades son más importantes. Con las funcionalidades priorizadas, se puede confeccionar un plan de

incrementos, donde en cada incremento se indica un subconjunto de funcionalidades que el sistema entregará. La asignación de funcionalidades a los incrementos depende de la prioridad dada a los requisitos. Finalizado el plan de incrementos, se puede comenzar con el primer incremento.

2.4.1.2 Modelo Iterativo

Este modelo, también conocido como Evolutivo, es una derivación del ciclo de vida en cascada puro, que busca reducir el riesgo que surge entre las necesidades del Usuario y el Producto final por malos entendidos durante la etapa de solicitud de requerimientos.

En el ciclo de vida iterativo, en cada Iteración se reproduce el ciclo de vida en cascada a menor escala. Los objetivos de una Iteración se establecen en función de la evaluación de las Iteraciones precedentes. Desde el principio, al final de cada Iteración se le entrega al Cliente una versión completa y mejorada del Producto. El Cliente es quien luego de cada Iteración evalúa el Producto y lo corrige o propone mejoras.[5]

Estas Iteraciones irán Refinando el sistema y se repetirán hasta obtener un Producto que satisfaga al Cliente.

La Especificación de requisitos se realiza en forma creciente: a medida que los Usuarios logran un mejor entendimiento del problema, éste es reflejado en el sistema software. Es decir, el Producto de cada etapa de Especificación de requisitos es un agregado o mejora al Producto de la etapa de especificación anterior.[5]

Este modelo se basa en dos premisas:

- Los Usuarios a menudo no saben bien lo que quieren o necesitan.
- Por lo general, los requisitos en algún momento van a cambiar.

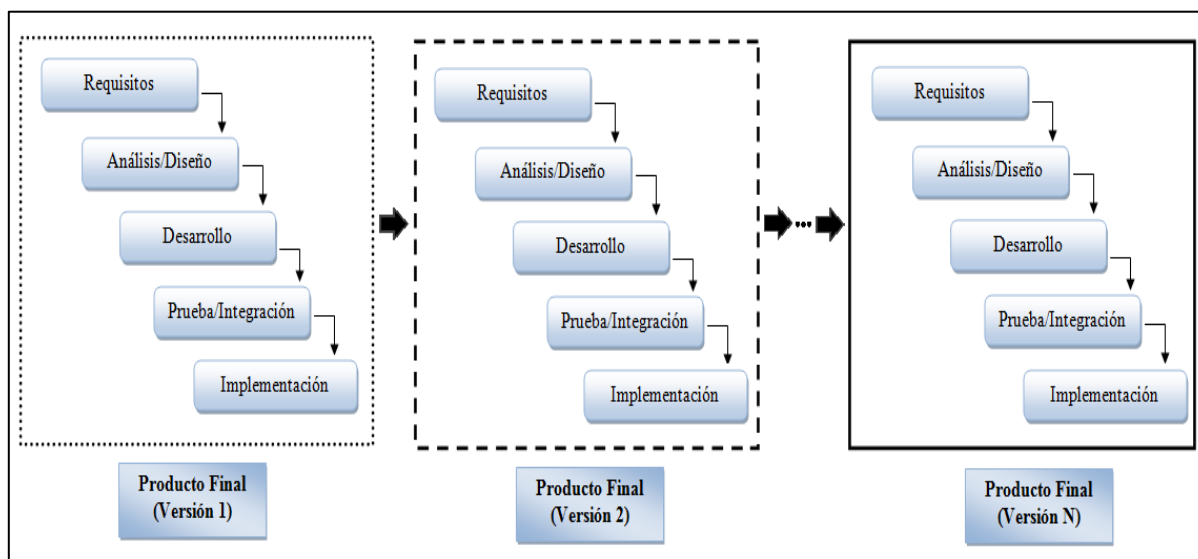
Para solucionar el primer punto, los requisitos se determinan en base a alguna forma operacional del sistema (por ejemplo, un prototipo) para ser revisado por los Usuarios. Para atender el segundo punto, se realizan entregas parciales del sistema que permiten incorporar nuevos requisitos o cambios en requisitos existentes en la siguiente entrega. Es decir, cada versión es una mejora sobre la predecesora.

Este modelo se utiliza cuando no se puede especificar a priori “todos” los requisitos del

software, sino que el proceso ayudará a ir descubriendo paso a paso los requisitos a partir de cada nueva Entrega.

Figura N° 2. 3

Modelo Iterativo



Fuente: [5, Fig. 1]

2.4.1.3 Modelo Prototipado

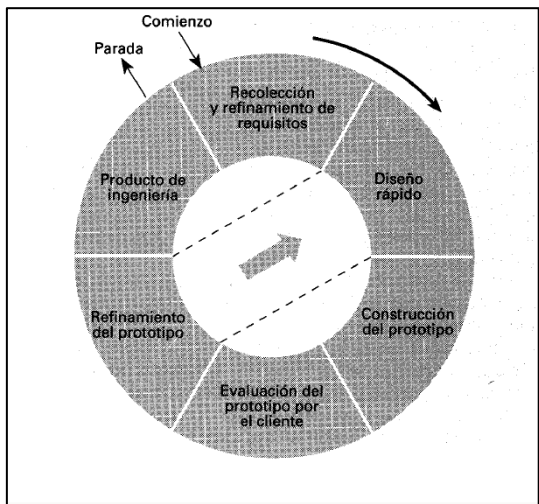
Es frecuente que un cliente defina un conjunto de objetivos generales para el software, pero que no identifique los requerimientos detallados para las funciones y características. En otros casos, el desarrollador tal vez no esté seguro de la eficiencia de un algoritmo o de la adaptabilidad de un sistema operativo o de la forma que debe adoptar la interacción entre el humano y la máquina. [10]

Analistas y clientes se reúnen para definir los objetivos generales del software, identifica cualesquiera requerimientos que conozca y detecta las áreas en las que es imprescindible una mayor definición. Se planea rápidamente una iteración para hacer el prototipo y se lleva a cabo el modelado (en forma de "diseño rápido"). Este se centra en la presentación de aquellos aspectos del software que serán visibles para los usuarios finales (interfaz o formatos de salida). El diseño rápido lleva a la construcción de un prototipo. Este se entrega y es evaluado por los

participantes, que dan retroalimentación para mejorar los requisitos. Las iteraciones se suceden a medida que el prototipo es refinado para satisfacer las necesidades del cliente.

Figura N° 2. 4

Ciclo de modelo prototipado



Fuente: [10, Fig. 1]

2.4.1.4 Selección de metodología

Para realizar la selección de una metodología es necesario conocer cada uno de ellos y comparando cual es el que se adaptará al proyecto.

Tabla 2. 2

Comparativa de modelos de desarrollo de software

Modelo	Características	Ventajas	Desventajas
Modelo Iterativo	Proporciona soporte para establecer la efectividad de los procesos y	Reduce necesidades del usuario y el producto final.	La entrega temprana de los proyectos produce la creación de sistemas demasiados simples que a veces se ven un poco monótonos a los ojos

Modelo Iterativo	de la calidad del producto.	Permite separar la complejidad del proyecto, esto gracias a su desarrollo por parte de cada iteración o bloque.	del personal que lo recibe.
Modelo Iterativo	El producto es consistente y puntual en el desarrollo.		El trato con el cliente debe basarse en principios éticos y colaboración mutua, más que trabajar cada parte independientemente, defendiendo sólo su propio beneficio.
Prototipado	Describe las fases principales de desarrollo de software. Define las fases primarias esperadas de ser ejecutadas durante esas fases	Útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. Código reutilizable	A modo de crear buenos prototipos se desatienden aspectos importantes como la calidad. Hacer pensar que el producto final está terminado. Cambios excesivos en el proyecto.
Incremental	Permite incrementos pequeños. Es sencillo manejar las tareas en cada iteración.	Se reduce el tiempo de desarrollo inicial. Se puede entregar versiones tempranas al cliente Divide el proyecto en partes más pequeñas.	Requiere mucha planeación tanto técnica y administrativa. Se necesita metas claras para el desarrollo del proyecto.

	Se adapta a necesidades que vayan surgiendo.		
--	--	--	--

Fuente: [Elaboración propia]

En el presente proyecto se utilizará el modelo incremental, por que ayuda a realizar una planificación anticipada para el desarrollo de software, y se tiene módulos del proyecto terminados en cada incremento.

2.4.2 UML

Es una de las herramientas más emocionantes en el mundo actual en el desarrollo de sistemas. Esto se debe a que los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlás a otras personas. [11]

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, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar 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 gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional, Rational Unified Process o RUP), pero no especifica en sí mismo qué metodología o proceso usar [11].

2.4.2.1 Versiones

Los antecedentes de UML se sitúan en la década de los 90 con distintos estándares para modelado de software, no obstante, podemos hablar de dos grandes versiones:

UML 1.X (UML 1.1-1.5): Desde finales de los 90 se empezó a trabajar con el estándar UML.

En los años sucesivos fueron apareciendo nuevas versiones que introducían mejoras o ampliaban a las anteriores.

UML 2.X (comprende UML 2.1 hasta UML 2.5, 2.6, etc.): En torno a 2005 se difundió una nueva versión de UML a la que podemos denominar UML 2.X. Comprenden varias revisiones.

UML 3.X: Evolución que se espera para UML 2.X.

2.4.2.2 Diagramas

Usando UML se pueden construir numerosos tipos de diagramas. Vamos a citar algunos:

- **Diagramas de casos de uso**

Representan a los actores y casos de uso (procesos principales) que intervienen en un desarrollo de software.

- **Diagramas de clases**

Para UML una clase es una entidad, no una clase software. Un diagrama de clases UML puede ser un diagrama del dominio o representación de conceptos que intervienen en un problema, o también un diagrama de clases software. El sentido de un diagrama UML se lo da la persona que lo construye.

- **Diagramas de secuencia**

Suelen usarse para representar objetos software y el intercambio de mensajes entre ellos, representando la aparición de nuevos objetos de izquierda a derecha.

- **Diagramas de colaboración**

Suelen usarse para representar objetos o clases y la forma en que se transmiten mensajes y colaboran entre ellos para cumplir un objetivo.

- **Diagramas de estados**

Suelen usarse para representar cómo evoluciona un sistema (cómo va cambiando de estado) a medida que se producen determinados eventos.

2.4.3 Selección de versión de UML

Se usará la versión 2 porque tiene varias revisiones y además esta incluye los diagramas que son requeridas y necesarias en el proyecto.

2.5 Lenguajes de programación

Los lenguajes de programación son bastante usados por la comunidad de desarrolladores de software y estas sirven para crear distintos tipos de sistemas.

2.5.1 PHP

Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas. Un sitio con páginas dinámicas es el que permite interactuar con el visitante, de modo que cada usuario que visita la página vea la información modificada para requisitos particulares. Las aplicaciones dinámicas para el Web son frecuentes en los sitios comerciales (e-commerce), donde el contenido visualizado se genera de la información alcanzada en una base de datos u otra fuente externa.[12]

2.5.1.1 Soporte para bases de datos:

Una de sus características más potentes es su soporte para gran cantidad de bases de datos. Entre su soporte pueden mencionarse InterBase, mSQL, MySQL, Oracle, Informix, PostgreSQL, entre otras. PHP también ofrece la integración con las varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en PDF hasta analizar código XML.[12]

2.5.2 Python

Python es un lenguaje de programación de alto nivel, interpretado y usado en varias áreas. En los últimos años su utilización ha ido constantemente creciendo y en la actualidad es uno de los lenguajes de programación más empleados para el desarrollo de software. [13]

Empresas y organizaciones del calibre de Industrial Light & Magic, Walt Disney, la NASA, Google, Yahoo!, Red Hat y Nokia hacen uso intensivo de este lenguaje para desarrollar sus productos y servicios. Esto demuestra que Python puede ser utilizado en diversos tipos de sectores, con independencia de su actividad empresarial.

Entre las principales razones para elegir Python, son muchos los que argumentan que sus principales características lo convierten en un lenguaje muy productivo. Se trata de un lenguaje potente, flexible y con una sintaxis clara y concisa. Además, no requiere dedicar tiempo a su compilación debido a que es interpretado. [13]

2.5.3 C#

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando estos últimos años con el objetivo de mejorar tanto su sistema operativo como su modelo de componentes para obtener una plataforma con la que sea sencillo el desarrollo de software en forma de servicios web. [14]

C# combina los mejores elementos de variados lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue igualmente el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++.

La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los programadores de C/C++/Java sea lo más inmediata posible. [14]

2.5.4 Selección de lenguaje de programación

Tabla 2. 3

Comparativa de lenguajes de programación

Lenguaje	Características	Ventajas	Desventajas
PHP	Gran documentación. Permite programación orientada a objetos. Módulos externos para mejorar la aplicación web.	Es un lenguaje multiplataforma. Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a	Revisar detalladamente los protocolos de seguridad para no dejar brechas de seguridad comprometidas. Uso recomendado de framework.

PHP	<p>Su uso de PHP es libre y también multi-plataforma.</p>	<p>información almacenada en una Base de Datos.</p> <p>El código fuente escrito en PHP y no es visible al navegador.</p> <p>Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. Tiene bastante documentación y una gran comunidad.</p>	
C#	<p>Sencillez de uso.</p> <p>Modernidad.</p> <p>Orientado a objetos.</p> <p>Orientado a componentes.</p> <p>Fuertemente tipado.</p>	<p>Declaraciones en el espacio de nombres: al empezar a programar algo, se puede definir una o más clases dentro de un mismo espacio de nombres.</p> <p>Tipos de datos: en C# existe un rango más amplio y definido de tipos de datos.</p> <p>Cada miembro de una clase tiene un atributo de acceso del tipo</p>	<p>Mantener actualizado Visual Studio .NET.</p> <p>Algunos requerimientos mínimos del sistema para poder trabajar adecuadamente.</p>

C#		público, protegido, interno y privado.	
Python	Open Source. Lenguaje Orientado a Objetos. Lenguaje de Alto Nivel. Bastantes librerías disponibles. Sintaxis clara.	Desarrollo más veloz: Se puede escribir un programa, salvarlo y ejecutarlo. En un lenguaje compilado tienes que pasar por los pasos de compilar y ligar el software, lo cual puede ser un proceso lento. Multiplataforma. Funciona en cualquier arquitectura, la única condición es que disponga del intérprete del lenguaje.	Al agregar más librerías este llega a ser lento.

Fuente: [Elaboración propia]

Se utilizará el lenguaje de programación PHP porque es un lenguaje que tiene una extensa comunidad, es libre y multiplataforma además que cuenta con un repositorio de librerías para su uso.

2.6 Entorno de trabajo

2.6.1 Laravel

El framework es un entorno de trabajo sobre la cual se puede crear distintos tipos de aplicaciones, en la actualidad existen una infinidad de ellos, pero a continuación se hará una comparación de los mejores y más usados.

Laravel es un nuevo y poderoso Framework PHP desarrollado por Taylor Otwell, que promete llevar al lenguaje PHP a un nuevo nivel.

Desarrollar aplicaciones usando Laravel es muy sencillo, fundamentalmente debido a su expresiva sintaxis, sus generadores de código, y su ORM incluido de paquete llamado Eloquent. [15]

Laravel, propone una forma de desarrollar aplicaciones web de un modo mucho más ágil. Por ejemplo, en Laravel opcionalmente es posible usar el patrón de diseño MVC (Modelo-Vista-Controlador) tradicional, donde al igual que otros frameworks PHP, el controlador es programado como una clase.

Por lo tanto, un Controlador es una clase PHP que dispone de métodos públicos que son el punto de entrada final de una petición HTTP³ a la aplicación. Pero este propone además una forma distinta y más directa de responder a la solicitud HTTP.

Laravel entrega la opción de seguir usando la metodología tradicional MVC. Sin embargo, el framework propone una vía más rápida en PHP, la cual consiste en programar la interacción HTTP directamente como una función anónima asociada a una Ruta.

Esto tiene la ventaja de reducir la cantidad de código, especialmente cuando sólo es necesario incluir una funcionalidad.

Así, donde antes era necesario programar una clase, ahora en Laravel sólo requiere escribir una función en PHP.

Por ejemplo, para desarrollar una aplicación que responda a la url: “http://mi-aplicacion.com/usuario/listar”. En el patrón MVC clásico es necesario crear un controlador

³ Http: Es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

“usuario” y la acción “listar” en cambio en Laravel con el manejador de rutas es más sencillo hacer el uso de MVC tradicional.

2.6.1.1 Manejo de los datos en Laravel

Laravel incluye una valiosa pieza de software, llamada Eloquent.

Este ORM se funda en patrón active record y su funcionamiento es extremadamente sencillo.

Un ORM (Object Relational Mapper) en PHP es un software que permite tratar la capa de persistencia de los datos, como simples accesos a métodos de una Clase u Objeto en PHP. La funcionalidad interna del ORM es mapear los objetos de PHP a las tablas en la base de datos, para el caso en que la persistencia de los datos de la aplicación es proporcionada por una DB. [15]

En Laravel es opcional el uso de Eloquent, pues también dispone de otros recursos que nos facilitan interactuar con los datos, o específicamente la creación de modelos.

La forma de interactuar con los datos en un patrón de diseño MVC, es mediante la creación de Modelos. Los Modelos son clases en PHP que encapsulan todo el trabajo con los datos de una aplicación. [15]

2.6.1.2 Modelos

Laravel usa el paradigma de programación donde se favorece "la convención sobre la configuración". Por ejemplo, Si es necesario obtener la lista de libros de la URL: “http://mi-aplicacion.com/libro/listar” en Laravel con una línea de código es posible interactuar con los modelos de datos y llamar distintos métodos.[15]

Las Vistas en Laravel, al igual que en otros framework PHP, son archivos de texto plano, que contienen una plantilla HTML junto con código PHP para desplegar la interfaz web al usuario de la aplicación.

2.6.1.3 Vistas

Laravel incluye de paquete un sistema de procesamiento de plantillas llamado Blade. Este sistema de plantillas, Blade favorece un código mucho más limpio en las Vistas, además de incluir un sistema de Caché que lo hace mucho más rápido.

Los Sistemas de Cache, evitan el tener que procesar el código una y otra vez en cada petición. Para lo cual, estos sistemas generan versiones estáticas en memoria o disco duro con archivos que corresponden a peticiones previamente procesadas. Y con esta técnica se logra mejorar el rendimiento de la aplicación. [15]

El sistema Blade de Laravel, permite una sintaxis mucho más reducida en su escritura. Lo cual, no es una gran ventaja sobre todo cuando siempre es posible usar una expresión resumida en PHP. No obstante, lo que, si es una gran ventaja, es el modo en que Blade maneja los layouts.

Los layouts⁴ en los frameworks PHP, permiten organizar las vistas en PHP. En especial, todos los elementos estáticos en una aplicación web que no cambian entre peticiones HTTP, como lo son: Header, Menús y Footers. [15]

Generalmente, a estos elementos se les denomina vistas parciales.

2.6.1.4 Blade

Los layouts en Blade, son archivos de texto plano que contiene todo el HTML de la página con etiquetas que representan elementos o zonas a incluir en las vistas parciales como se conocen en otros Frameworks en PHP.

Sin embargo, en Blade estos elementos incrustados se organizan en un sólo archivo. Esta es una idea muy interesante de Laravel que mejora la organización de las vistas y su rendimiento. Sobre todo, cuando las vistas pueden llegar a ser muy complejas incluso con elementos anidados.

En el render de una Vista completa en Laravel se usan dos (2) archivos:

- El layout definiendo el HTML global y las zonas a incluir.
- Un sólo archivo, la Vista, con los elementos (partial views).

En el layout presentado, el código `@yield()` identifica al método donde como parámetro se indica el nombre de la zona desplegar. Por otro lado, el código de la vista, donde se define el layout a usar y la información de las distintas zonas a desplegar.

⁴ Layout: Sirve para hacer referencia al esquema que será utilizado y cómo están distribuidos los elementos y formas dentro de un diseño.

2.6.2 Symfony

Es uno de los framework más completos, desarrollado por la empresa SensioLab⁵, está diseñado para optimizar el desarrollo de aplicaciones web. Este framework separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web. También mediante el uso de varias herramientas reduce el tiempo de desarrollo de una aplicación web compleja. [16]

A parte de esto, para el desarrollador supone una gran ventaja su uso, ya que, al automatizar las tareas más comunes, dicho desarrollador puede dedicarse por completo a los aspectos específicos de cada aplicación. [16]

Symfony está completamente desarrollado en PHP y es compatible con la mayoría de los gestores de base de datos, y además se puede ejecutar tanto en plataformas nix, como pueden ser Linux o Unix, como en plataformas Windows.

2.6.3 Yii2

Yii es un framework de PHP de alto rendimiento, basado en componentes para desarrollar aplicaciones web modernas en poco tiempo. El nombre Yii significa "simple y evolutivo" en chino. También se puede considerar como un acrónimo de Yes It Is (que en inglés significa Sí, ¡eso es)! [17].

Yii es un framework genérico de programación web, lo que significa que se puede utilizar para desarrollar todo tipo de aplicaciones web en PHP. Debido a su arquitectura basada en componentes y a su sofisticada compatibilidad de caché, es especialmente apropiado para el desarrollo de aplicaciones de gran envergadura, como portales, foros, sistemas de gestión de contenidos (CMS), proyectos de comercio electrónico, servicios web compatibles con la arquitectura REST y muchas más. [17]

2.6.3.1 Usabilidad

Como la mayoría de los framework de PHP, Yii implementa el patrón de diseño MVC (Modelo-Vista-Controlador) y promueve la organización de código basada en este patrón.

⁵ SensioLab: Compañía francesa que provee consultoría, servicios, formación sobre tecnologías open source.

La filosofía de Yii consiste en escribir el código de manera simple y elegante, sin sobre diseñar nunca por el mero hecho de seguir un patrón de diseño determinado.

Yii es un framework completo que provee muchas características probadas y listas para usar, como los constructores de consultas y la clase ActiveRecord⁶ para las bases de datos relacionales y NoSQL, la compatibilidad con la arquitectura REST para desarrollar API, la compatibilidad de caché en varios niveles y muchas más. [17]

Yii es extremadamente extensible. Puedes personalizar o reemplazar prácticamente cualquier pieza de código de base, como se puede también aprovechar su sólida arquitectura de extensiones para utilizar o desarrollar extensiones distribuibles.

El alto rendimiento es siempre la meta principal de Yii.

2.6.4 Selección de entorno de trabajo

A continuación, se mostrará las ventajas y desventajas de cada framework mencionado anteriormente.

Tabla 2. 4

Comparativa de Entornos de trabajo

Framework	Características	Ventajas	Desventajas
Laravel	<p>Sistema de ruteo.</p> <p>Mapeado de base de datos integrado.</p> <p>Incluye motor de vistas Blade.</p> <p>Amplia comunidad.</p> <p>Bastante documentación.</p>	<p>Desarrollo rápido de aplicaciones.</p> <p>Agregar diversas librerías externas.</p> <p>Curva baja de aprendizaje.</p> <p>Escalable.</p>	<p>Ciertas configuraciones en el proyecto para publicar en los servidores.</p> <p>Requiere de últimas versiones de PHP.</p>

⁶ActiveRecord: Patrón de diseño que define una forma de acceder a los datos de una base de datos relacional y convertir las filas de una tabla en objetos.

Laravel	<p>Incluye CLI artisan para crear nuevos componentes en el proyecto.</p> <p>El framework más popular.</p> <p>Passport integrado.</p>		
Symfony 2	<p>Framework orientado a MVC.</p> <p>ORM compatible con diferentes bases de datos.</p> <p>Orientado para aplicaciones empresariales.</p> <p>Separación de Lógica de servidor y presentación.</p>	<p>Independiente de la base de datos.</p> <p>Librerías preconfiguradas.</p>	<p>Únicamente usada para proyectos grandes.</p> <p>Curva baja de aprendizaje.</p>
Yii 2	<p>Usa el patrón MVC.</p> <p>Integración con Ajax.</p> <p>Integración con Zend framework.</p> <p>Autenticación Integrada.</p>	<p>Cuenta con una biblioteca de extensiones llamada zii.</p> <p>Herramienta de código integrado gii.</p> <p>Generador de documentación.</p>	<p>Mayor uso de memoria.</p>

Fuente: [Elaboración propia]

Por las características y ventajas que tiene Laravel, será la herramienta para desarrollar el proyecto.

2.7 Arquitectura

La arquitectura de software define la forma de trabajar un sistema, como construir nuevos módulos, y también deja intuir el tipo de aplicación que describe.

2.7.1 Arquitectura monolítica

Al momento de crear una aplicación es importante tener en claro hasta donde va a llegar o que tantos recursos vamos a implementar para desarrollarla, existen distintos factores que se toman en cuenta las que vamos a describir a continuación.

Al principio una aplicación puede ser pequeña y puede tener simples requerimientos, pero conforme pasa el tiempo esta va creciendo y los requerimientos van aumentando y es donde entra en juego que tipo de arquitectura va a ser usada.

Una arquitectura monolítica no depende de servicios externos fuera de esta, solo obedece a sus componentes de su interior.

2.7.1.1 Escalabilidad vertical

En una arquitectura monolítica la eficiencia puede aumentar la deuda técnica. Como, por ejemplo, tareas con resolución de errores, modificaciones de la interfaz, cambios que afectaran a la aplicación en general, las bases de código monolítica consumen más tiempo y son menos adaptables y más caras de mantener.

Una de las características de la arquitectura es que son escalables verticalmente lo que significa que crecer en hardware, es decir que a medida que crezca la aplicación monolítica se necesitara más recursos como ser: más memoria, mejor procesador, etc.

Tarde o temprano esta aplicación tendrá un punto limite

2.7.2 Arquitectura orientada a servicios

Arquitectura orientada al servicio (SOA) es un enfoque de diseño, donde múltiples servicios colaboran para proporcionar un conjunto final de las capacidades. Un servicio aquí normalmente significa un proceso del sistema operativo completamente independiente. SOA surgió como un

enfoque para combatir los retos de las grandes aplicaciones monolíticas. Es un enfoque que tiene como objetivo promover la reutilización de software [19].

Una característica importante de esta es el escalamiento horizontal.

2.7.2.1 Escalabilidad horizontal

Sin duda este tipo de escalabilidad es el más potente, pero también es más complicado. Este modelo indica tener distintos servidores (conocidos como nodos) trabajan todos como uno, se crea una red de servidores, cuando el performance se ve afectado se agrega otro nodo. Esto garantiza una alta disponibilidad del sistema.

2.7.3 Microservicios

Es una arquitectura que va emergiendo y tomando fuerza con características interesantes haciendo frente a las clásicas arquitecturas monolíticas, definiendo “el principio de individualidad” o “divide y vencerás”.

Sam Neumann indica que los Microservicios son pequeños servicios, autónomos que trabajan en conjunto. [18].

Por otra parte, Esaú A. indica que la arquitectura de microservicios o microservicios a secas es un distintivo sistema de desarrollo de software que ha crecido en popularidad en los últimos años. De hecho, a pesar de que su extensión en uso no ha llegado tan allá donde sí lo ha hecho su teoría, muchos desarrolladores están descubriendo cómo esta forma de creación de software favorece el tiempo, rendimiento y estabilidad de sus proyectos [18].

A continuación, se define algunas características.

2.7.3.1 Autónomo

Un microservicio es una entidad separada. Puede ser desplegado como un servicio aislado en una plataforma como servicio (PaaS).

Todas las comunicaciones entre los servicios que ellos mismos son a través de llamadas de red, para hacer cumplir la separación entre los servicios y evitar los peligros de acoplamiento fuerte. [18]

Los servicios exponen las interfaces de programación de aplicaciones (API) y las comunicaciones se hacen a través de ellas.

Sin desacoplamiento todo se descompone para.

Los beneficios de este son varios en los que pueden imputarse a la aplicación de cualquier sistema externo.

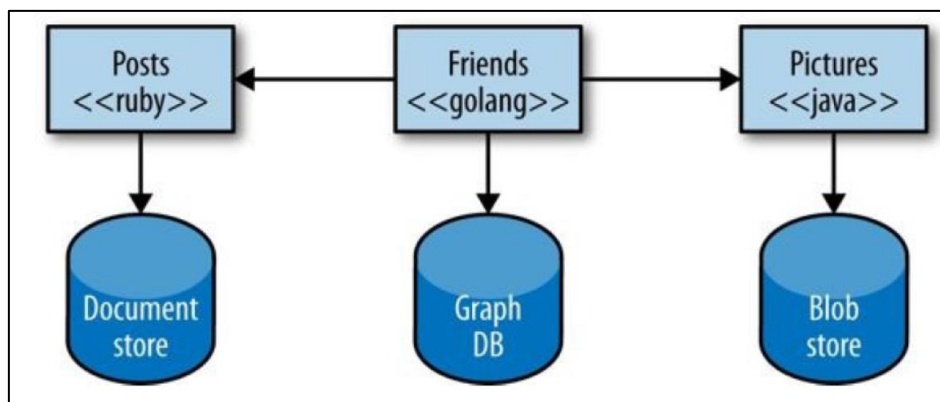
Se puede utilizar diferentes tecnologías que colaboraran con la arquitectura. Esto nos adecua usar una herramienta para cada trabajo en lugar de estandarizarlo en unos solo.

Con microservicios, también somos capaces de adoptar una tecnología más rápida, y entender cómo los nuevos avances nos pueden ayudar. Uno de los mayores obstáculos para probar y la adopción de la nueva tecnología es los riesgos asociados a ella.

Con una aplicación monolítica, si se quiere probar un nuevo lenguaje de programación, bases de datos o marco de trabajo, cualquier cambio afectará a una gran cantidad del sistema. Con un sistema que consta de múltiples servicios, teniendo varios nuevos lugares en los que probar una nueva pieza de tecnología. Se puede elegir un servicio que es quizás menor riesgo y el uso de la tecnología allí, sabiendo que se puede limitar cualquier posible impacto negativo. Muchas organizaciones consideran esta capacidad de absorber más rápidamente nuevas tecnologías para ser una ventaja real para ellos [18].

Figura N° 2. 5

Arquitectura Microservicio



Fuente: [18, Fig. 5]

2.7.3.2 Prioritario

La forma en la que se organizan los microservicios suele ser en torno a las necesidades, capacidades y prioridades del cliente o negocio en el que se implantará. A diferencia de un entorno monolítico donde cada equipo de trabajo tiene un enfoque específico sobre un apartado de la aplicación, en la arquitectura de microservicios se utilizan módulos multifuncionales, adaptando así un módulo común a todos para que ofrezca un servicio determinado. El ahorro en

tiempo de desarrollo es inmenso, por no hablar de la comodidad a la hora de programar tareas de mantenimiento, donde podemos revisar un módulo mientras el resto del equipo de trabajo no ve interrumpida su jornada. [18]

2.7.3.3 Tolerancia a fallos

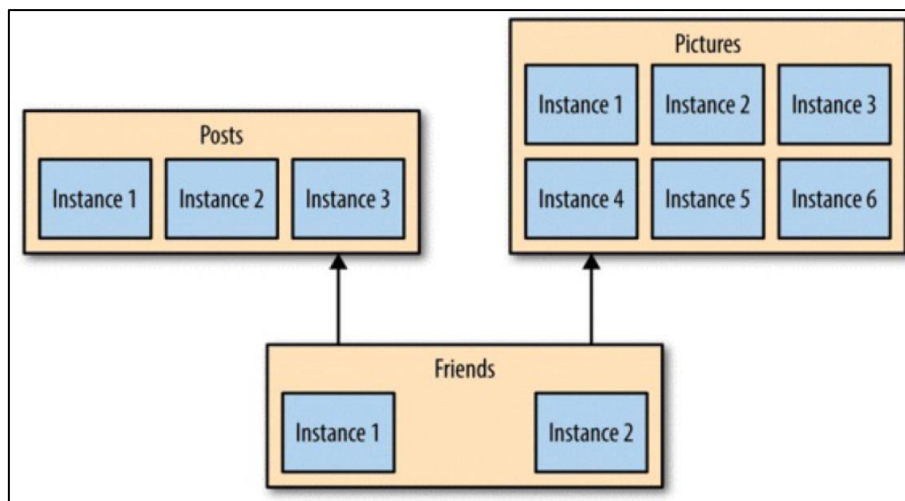
Un concepto clave en la ingeniería de la resiliencia es la mampara. Si uno de los componentes de un sistema falla, pero que los fallos no se conectan en cascada, se puede aislar el problema y el resto del sistema puede seguir trabajando. los límites del servicio se convierten en sus mamparos obvias. En un servicio monolítico, si el servicio falla, todo deja de funcionar. Con un sistema monolítico, que puede ejecutarse en múltiples máquinas para reducir nuestras posibilidades de fracaso, pero con microservicios, podemos construir sistemas que manejan el fracaso total de los servicios y degradar la funcionalidad en consecuencia. [18]

2.7.3.4 Gestión de Despliegues

Con un servicio grande, monolítico, se tiene que escalar todo junto. Una pequeña parte de nuestro sistema global está limitado en el rendimiento, pero si ese comportamiento es encerrado en una aplicación monolítica gigante, tenemos que manejar escalamiento todo como una pieza. Con los servicios más pequeños, se puede escalar los servicios que estos necesitan, lo que nos permite ejecutar otras partes del sistema en un hardware más pequeño, menos potentes.

Figura N° 2. 6

Orientado solo a los que necesitan



Fuente: [18, Fig. 6]

2.7.3.5 Facilidad de implementación

Un cambio de una línea para una aplicación monolítica millones de líneas de larga requiere toda la aplicación para ser desplegado a fin de liberar el cambio. Eso podría ser un gran impacto, el despliegue de alto riesgo. En la práctica, de gran impacto, las implementaciones de alto riesgo terminan sucediendo con poca frecuencia debido al temor comprensible. Desafortunadamente, esto significa que nuestros cambios se acumulan y se acumulan entre los lanzamientos, hasta que la nueva versión de nuestra producción golpear aplicación tiene masas de cambios. Y cuanto mayor sea el delta entre versiones, ¡mayor es el riesgo de que vamos a llegar algo mal!

Con microservicios, podemos hacer un cambio a un único servicio y desplegarlo de forma independiente del resto del sistema. Esto nos permite obtener nuestro código desplegado más rápido. Si se produce un problema, se puede aislar rápidamente a un servicio individual, haciendo rápida reversión fácil de lograr. También significa que podemos conseguir nuestra nueva funcionalidad a los clientes más rápido. Esta es una de las razones principales por las organizaciones como Amazon y Netflix utilizan estas arquitecturas – para asegurarse de que eliminar el mayor número posible de impedimentos para conseguir el software fuera de la puerta [18].

2.7.4 Selección de arquitectura

Se listará las ventajas y desventajas de cada arquitectura mencionada anteriormente

Tabla 2. 5

Comparativa de arquitecturas

	Características	Ventajas	Desventajas
Microservicios	Autónomo. Tolerante a fallos. Gestión de despliegue. Fácil implementación. Fácil mantenibilidad por módulo. Versionamiento modular.	Escalabilidad. Funcionalidad modular, módulos independientes. Libertad del desarrollador de desarrollar y desplegar	Alto consumo de memoria. Necesidad de tiempo para poder dividir distintos microservicios. Pruebas o testeos complicados al

Microservicios		servicios de forma independiente. Uso de contenedores permitiendo el despliegue y el desarrollo de la aplicación rápidamente.	despliegue distribuido.
Monolítica	Recomendado para pequeñas aplicaciones.	Fácil implementación. Solución rápida y económica para pequeñas aplicaciones.	Crecimiento limitado por hardware. Una falla en el servidor provoca que se detenga toda la aplicación. No soporta alta disponibilidad.

Fuente: [Elaboración propia]

Se eligió la arquitectura de microservicios porque representa una facilidad en despliegue y presenta una mayor disponibilidad del sistema.

2.8 Servicios Web

Un servicio web es un conjunto de protocolos y/o estándares que son utilizados para intercambiar datos entre distintas aplicaciones.

2.8.1 REST

REST cambió por completo la ingeniería de software a partir del año 2000. Esta nueva orientación de desarrollo de proyectos y servicios web fue determinado por **Roy Fielding**, el padre de la especificación HTTP y uno de los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación “Architectural Styles and the Design of Network-

based Software Architectures”. En el campo de las APIs, REST (Representational State Transfer- Transferencia de Estado Representacional), actualmente es el alfa y omega del desarrollo de servicios de aplicaciones. [20]

En la actualidad hay cientos de empresas que generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos que disponen de una gran capacidad, pero también mucha complejidad. En ocasiones es optar una solución más sencilla de manipulación de datos como REST.

2.8.1.1 Características

Protocolo cliente/servidor sin estado: cada petición HTTP tiene la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable. [19]

Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

- Los objetos en REST siempre se manipulan a partir de la URI. Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.
- Interfaz uniforme: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén

identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

- Sistema de capas: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
- Uso de hipermedios: hipermedia es un término acuñado por Ted Nelson en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

A continuación, un ejemplo de una solicitud REST en formato Json.

Figura N° 2. 7

Formato Json

```
{  
  "id": 78,  
  "nombre": "Ronald",  
  "productos": [  
    {"producto1": "http://miservidor/stock/api/v1/cementoPortland/1/"}  
  ]  
}
```

Fuente: [Elaboración propia]

2.8.2 SOAP

Es un protocolo derivado de XML que sirve para intercambiar la información entre aplicaciones. En estas se pueden identificar dos tipos de mensaje según su contenido.

Mensajes orientados al documento: Contienen cualquier tipo de contenido que queramos enviar entre aplicaciones. [20]

Mensajes orientados a RPC: Este tipo de mensajes sirve para invocar procedimientos de forma remota (Remote Procedure Calls). Se puede ver como un tipo más concreto dentro del tipo anterior, ya que en este caso como contenido del mensaje especificaremos el método que queremos invocar junto a los parámetros que le pasamos, y el servidor deberá devolver como respuesta un mensaje SOAP con el resultado de invocar el método.

- **WSDL**

Es un lenguaje basado en XML utilizado para describir la funcionalidad que proporciona un servicio Web. Una descripción WSDL (fichero WSDL) de un servicio web proporciona una descripción entendible por la máquina de la interfaz del servicio Web, indicando cómo se debe llamar al servicio, qué parámetros espera, y qué estructuras de datos devuelve. [20]

- **UDDI**

Permite localizar Servicios Web. Para ello define la especificación para construir un directorio distribuido de Servicios Web, donde los datos se almacenan en XML. En este registro no sólo se almacena información sobre servicios, sino también sobre las organizaciones que los proporcionan, la categoría en la que se encuentran, y sus instrucciones de uso (normalmente WSDL). [21]

2.8.3 Selección de Servicio Web

Tabla 2. 6

Servicio	Características	Ventajas	Desventajas
SOAP	Ficheros WSDL basado en XML. Múltiples instancias del proceso	Se puede depurar. Incrementa la privacidad	Transferencia de datos más lento. Descripción semántica informal.
REST	Uso alternativo de Json o XML.	Bajo consumo de recursos. Fácil implantación.	Se necesita puertos específicos.

Fuente: [Elaboración propia]

Se eligió el servicio web REST porque su uso permite devolver solamente datos requeridos, al contrario de SOAP permite el uso de XML y JSON, según la configuración de este.

2.9 Base de datos

Para el almacenamiento de base de datos se necesita un lugar donde se almacene, para eso se listará algunas bases de datos y se elegirá la que mejor se adapte al proyecto.

2.9.1 Mariadb

Mariadb es un sistema de gestión de base de datos derivado de Mysql con licencia (GPL). Fue desarrollado por Michael Widenius quien fue fundador de Mysql y actualmente la fundación MariaDB. [22]

Introduce dos motores de almacenamiento nuevos, uno llamado Aria -que reemplaza con ventajas a MyISAM y otro llamado XtraDB en sustitución de InnoDB. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente SQL Server. Todo esto surge por la compra de Sun Microsystems la que previamente compro MySQL AB, la compra se debe a que Oracle quiso reducir la competencia que MySQL suponía. [22]

2.9.2 SQL Server

Esta es una de las bases de datos más usadas en el mundo, teniendo gran aceptación por grandes empresas

SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) de Microsoft que está diseñado para el entorno empresarial. SQL Server se ejecuta en T-SQL (Transact -SQL). Tiene una interfaz gráfica la cual facilita de gran manera el uso de esta herramienta,

Tiene la capacidad de soportar grandes procesamientos almacenados, tiene una licencia de pago otorgado por Microsoft.

El principal medio de interacción con el servidor es T-SQL(Transact-SQL) para los clientes de Microsoft SQL Server 2005 en adelante. [23]

2.9.3 Selección de base de datos

Luego de haber mencionado las bases de datos se hará una descripción de ventajas y desventajas de cada una de las bases de datos.

Tabla 2. 7

Comparativa de base de datos

Base de Datos	Características	Ventajas	Desventajas
MariaDB	Nuevos motores más eficientes, Aria y XtraDB. Repara muchos errores en el código de MySQL. Soporte de procedimientos almacenados.	Ofrece más de 200.000 conexiones a MariaDB. La comunidad de desarrolladores sigue dueños de sus contribuciones. Open source.	No posee interfaz gráfica para su uso, es necesario recurrir a software de terceros.
SQL Server	Escalabilidad, estabilidad y seguridad. Soporte de transacciones. Soporta procedimientos almacenados.	Incluye un entorno gráfico de administración. Permite administrar información de otros servidores de datos.	Se debe pagar licencia por el uso. Usa demasiada memoria RAM para las transacciones.

Fuente: [Elaboración propia]

Se determino usar MariaDB por que representa una alternativa en base de datos. También es de código abierto lo que representa que no tiene costo por el uso.

2.10 Seguridad

Para que un sistema tenga seguridad y no sea vulnerable existen ciertos métodos para proteger la información, a continuación, se mencionara alguno de ellos.

2.10.1 Json Web Token

JSON Web Token (JWT) es un estándar abierto que define una forma compacta y autónoma para transmitir de forma segura información entre las partes como un objeto JSON. Esta información puede ser verificada y confiable porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves públicas / privadas usando RSA. [24]

Se explicará algunos conceptos de estas definiciones.

- **Compacto:** debido a su tamaño más pequeño, los JWT se pueden enviar a través de un URL, parámetro POST o dentro de un encabezado HTTP. Además, el tamaño más pequeño significa que la transmisión es rápida.
- **Autónomo:** la carga contiene toda la información requerida sobre el usuario, evitando la necesidad de consultar la base de datos más de una vez.

2.10.1.1 Usabilidad

Autenticación: este es el escenario más común para usar JWT. Una vez que el usuario haya iniciado sesión, cada solicitud posterior incluirá el JWT, lo que le permitirá acceder a las rutas, servicios y recursos permitidos con ese token. Single Sign On es una función que usa mucho JWT hoy en día, debido a su pequeña sobrecarga y su capacidad de ser utilizada fácilmente en diferentes dominios. [24]

Intercambio de información: los tokens web JSON son una buena forma de transmitir información de forma segura entre las partes. Como los JWT se pueden firmar, por ejemplo, usando pares de claves públicas / privadas, puede estar seguro de que los remitentes son quienes dicen ser. Además, como la firma se calcula utilizando el encabezado y la carga útil, también puede verificar que el contenido no haya sido alterado.

2.10.1.2 Estructura

Los tokens web JSON constan de tres partes separadas por puntos (.), que son:

- **Encabezamiento** El encabezado generalmente consta de dos partes: el tipo de token, que es JWT, y el algoritmo de hash que se utiliza, como HMAC SHA256 o RSA.
- **Carga útil** La segunda parte del token es la carga útil, que contiene los reclamos. Los reclamos son declaraciones sobre una entidad (generalmente, el usuario) y metadatos adicionales. Hay tres tipos de reclamaciones: reservadas, públicas y privadas.
- **Reclamaciones reservadas:** Se trata de un conjunto de notificaciones predefinidas que no son obligatorias, sino recomendadas, para proporcionar un conjunto de notificaciones útiles e interoperables. Algunos de ellos son: iss(issuer), exp (expiration time), sub (subject), aud (audiencia) y otros.
- **Reclamaciones públicas:** Pueden ser definidas a voluntad por aquellos que usan JWT. Pero para evitar colisiones, deben definirse en el Registro de tokens web IANA JSON o definirse como un URI que contenga un espacio de nombres resistente a colisiones.
- **Reclamos privados:** Estos son los reclamos personalizados creados para compartir información entre las partes que acuerdan usarlos.
- **Firmas** La tercera parte del JWT es la firma. Estas son usadas para cifrar la información y por medio de esta serán verificadas por el receptor.

2.10.1.3 Funcionamiento

En la autenticación, cuando el usuario inicia sesión con éxito usando sus credenciales, se devolverá un Token web JSON que se debe guardar localmente (generalmente en el almacenamiento local, pero también se pueden usar cookies), en lugar del enfoque tradicional de crear una sesión en el servidor y devolver una cookie.

Siempre que el usuario quiera acceder a una ruta o recurso protegido, el agente de usuario debe enviar el JWT, normalmente en el encabezado de autorización utilizando el esquema del Portador.

Este es un mecanismo de autenticación sin estado ya que el estado del usuario nunca se guarda en la memoria del servidor. Las rutas protegidas del servidor comprobarán si hay un JWT válido en el encabezado de Autorización, y si está presente, el usuario podrá acceder a los recursos

protegidos. Como los JWT son autónomos, toda la información necesaria está allí, lo que reduce la necesidad de consultar la base de datos varias veces.

Esto le permite confiar completamente en las API de datos que son apátridas e incluso realizar solicitudes a los servicios descendentes. No importa qué dominios están sirviendo a sus API, por lo que el Intercambio de recursos de origen cruzado (CORS) no será un problema ya que no utiliza cookies.

2.10.1.4 Uso

Hablemos de los beneficios de JSON Web Tokens (JWT) en comparación con Simple Web Tokens (SWT) y Security Assertion Markup Language Tokens (SAML).

Como JSON es menos detallado que XML, cuando se codifica, su tamaño también es más pequeño, lo que hace que JWT sea más compacto que SAML. Esto hace que JWT sea una buena opción para aprobar en entornos HTML y HTTP.

En términos de seguridad, SWT solo puede ser firmado simétricamente por un secreto compartido usando el algoritmo HMAC. Sin embargo, los tokens JWT y SAML pueden usar un par de claves públicas / privadas en forma de un certificado X.509 para la firma. Firmar XML con firma digital XML sin introducir agujeros de seguridad oscuros es muy difícil si se compara con la simplicidad de la firma de JSON.

Los analizadores JSON son comunes en la mayoría de los lenguajes de programación porque se asignan directamente a los objetos. Por el contrario, XML no tiene un mapeo natural de documento a objeto. Esto hace que sea más fácil trabajar con JWT que con las aserciones de SAML.

En cuanto al uso, JWT se usa a escala de Internet. Esto resalta la facilidad del procesamiento del JSON Web token en el lado del cliente en múltiples plataformas, especialmente en dispositivos móviles.

2.10.2. Sesiones

La conexión que se establece entre un usuario (el equipo cliente) y un servidor Web se denomina sesión, las sesiones pueden abarcar múltiples páginas Web.

El seguimiento se realiza mediante la administración del estado, la administración del estado es el proceso por el cual mantenemos la misma información a través de múltiples peticiones para las mismas o distintas páginas Web, al igual que las tecnologías basadas en Hypertext Transfer Protocol (HTTP), los formularios Web Forms no tienen estado, lo que significa que no indican automáticamente si las peticiones de una secuencia son todas del mismo cliente [25].

2.10.2.1 Cookies

Una cookie es un archivo creado por un sitio de Internet para almacenar información en el equipo, como, por ejemplo, las preferencias cuando se visita el sitio. Cuando un usuario ingresa a una página que usa cookies, el sitio puede pedirle a su navegador que guarde una o más cookies en el disco duro. [26]

En algunos navegadores, cada cookie es un pequeño archivo. Las cookies a menudo guardan la configuración de los sitios web, como el idioma preferido o la ubicación. Así, cuando se visita de nuevo al sitio, el navegador envía de nuevo las cookies que le pertenecen, lo que le permite presentar información personalizada en función de las necesidades del usuario.

Las cookies también pueden guardar información que identifique a un usuario determinado. Esta información personal, como por ejemplo nombre, correo electrónico, domicilio, domicilio, puesto de trabajo o número de teléfono, es la que se puede usar para identificarte o contactar contigo. Sin embargo, un sitio web solo tiene acceso a la información personal que se le proporciona. Por ejemplo, un sitio web no podrá determinar la dirección de correo electrónico a menos que tú la ingreses. Un sitio web tampoco podrá tener acceso a otra clase de información que haya en tu equipo. [26]

2.10.3. Selección de tipo de seguridad

Se procederá a listar las ventajas y desventajas de los tipos de seguridad que será utilizado en el proyecto.

Tabla 2. 8

Comparativa de seguridad web

	Características	Ventajas	Desventajas
--	------------------------	-----------------	--------------------

JWT	<p>No almacena datos en el servidor.</p> <p>Utiliza el formato Json.</p> <p>Cada cierto tiempo el token generado es distinto al anterior.</p> <p>Es enviada en la cabecera de cada consulta al servidor.</p>	<p>Almacena información en el payload que luego puede usarse en el navegador.</p> <p>Completa documentación en la página oficial.</p> <p>Las peticiones son más seguras.</p>	<p>Complejidad de uso.</p> <p>Si se carga demasiado el payload las peticiones serán un poco lentas</p>
Sesiones	<p>Puede recuperar datos de la cookie.</p> <p>El servidor genera un identificador único para el cliente.</p> <p>Caducan cuando no está en actividad.</p>	<p>Por medio de la cookie puede almacenar grandes datos de información.</p>	<p>Necesita cookies para el manejo de información básica del cliente.</p> <p>El uso de varias sesiones puede llegar a ser complicado el manejo.</p>

Fuente: [Elaboración propia]

Se hará uso de Json Web Token ya que este es recomendado para este tipo de arquitecturas.

2.11 Calidad de software

La calidad del software es una preocupación a la que se dedican muchos esfuerzos. Sin embargo, el software casi nunca es perfecto. Todo proyecto tiene como objetivo producir software de la mejor calidad posible, que cumpla, y si puede supere las expectativas de los usuarios.

2.11.1 Calidad

Es la aptitud de un producto o servicio para satisfacer las necesidades del usuario. Así también es la cualidad de todos los productos, no solamente de equipos sino también de programas.

En el desarrollo de software, la calidad de diseño acompaña a la calidad de los requisitos, especificaciones y diseño del sistema. La calidad de concordancia es un aspecto centrado principalmente en la implementación.

Si la implementación sigue al diseño, y el sistema resultante cumple con los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta.

2.11.2 Calidad de software

Es la que cumple los requisitos planteados en el inicio del desarrollo del software. No se puede medir la calidad de software de manera correcta. Por lo que se podría dar una medición aproximada a partir de la funcionalidad de los métodos y funcionalidades que tiene el software.

2.11.3 Pruebas unitarias

Es un método que puede hacer referencia a código para verificar la correcta funcionalidad y este mismo define si el resultado obtenido es el esperado. Estos métodos suelen ocupar poco tiempo para hacer las pruebas necesarias y pueden repetirse cuantas veces sea necesario.

2.11.4 Pruebas de integración

Las pruebas de integración comprueban que el sistema está funcionando. Acoplan partes del sistema y comprueban que encajan sin problemas. Las pruebas unitarias no tienen en cuenta elementos tan importantes como los accesos a base de datos o peticiones de red. No son suficiente para comprobar que el comportamiento es correcto.

Gracias a este tipo de pruebas se puede encontrar bugs en los pull requests sin necesidad de pasar regresivos relacionados con esa parte del producto. Hacen que el sistema sea fiable, pudiendo confiar en las partes protegidas por estas pruebas, ya que tendrán el comportamiento deseado al ser llamados por otros componentes. Protegen a los grupos de desarrolladores para que el código de otros componentes no les sorprenda de formas desagradables.

Al utilizar un lenguaje como Gherkin⁷ para escribir este tipo de pruebas se consigue además una documentación extra sobre el funcionamiento del sistema. Pudiendo saber rápidamente qué está cubierto por las pruebas y qué no. [27]

2.11.5 Pruebas funcionales

Es un paso más allá de las pruebas de integración y tratan de probar el sistema como lo haría un usuario. Aquí entra especialmente la automatización de interfaces gráficas. Son las pruebas funcionales que más mantenimiento necesitan y las más lentas.

Sin embargo, los beneficios son similares a los anteriores tipos de pruebas. Protegen contra interfaces fallidas, quitan trabajo a la hora de hacer pruebas manuales. Son especialmente útiles al configurarlos y prepararlos para ser ejecutados en todos los entornos en los que se distribuye o se ejecuta la aplicación. Por ejemplo, en los diferentes navegadores soportados, si el interfaz a probar fuera un entorno web.

Sin este tipo de pruebas se tendría que probar manualmente todas las interfaces de un determinado proyecto. Suelen ser los mejores para las pequeñas suites de smoke tests⁸. [28]

2.11.6 Selección de pruebas de software

Tabla 2.9
Comparativa de pruebas de funcionalidad

	Características	Ventajas	Desventajas
Pruebas Unitarias	Creación de métodos de prueba mediante código. Pueden repetirse cuantas veces sea necesario.	Implementación sencilla. Mayor seguridad de funcionamiento	Toma bastante tiempo implementar y probar cada método de prueba. Ejecutar método por método para probar la aplicación.

⁷ Gherkin: Define la estructura y una sintaxis básica para la descripción de las pruebas.

⁸ Smoke Tests: Pruebas que pretenden evaluar la calidad de un producto de software previo a una recepción formal.

Pruebas de Integración	Acoplan partes del sistema y encajan sin problemas.	Verifican que no haya errores en depuración. Se puede detectar fallas a etapas tempranas.	Solo verifican la integración entre módulos. Se debe esperar hasta que se termine el software.
Pruebas Funcionales	Las pruebas son como la haría el usuario final. Gran parte son pruebas manuales.	Protegen contra interfaces fallidas. Pruebas completas, como interfaz y datos.	Las pruebas manuales pueden ser más tediosas. Verificación de cada interfaz del sistema

Fuente: [Elaboración propia]

Después de ver los tipos de pruebas, se tomó la decisión de hacer pruebas funcionales al sistema ya que este garantiza una mejor calidad de software para el usuario final.

CAPÍTULO 3

MARCO PRÁCTICO

En el actual capítulo se desarrollará el modelo de negocio actual y alternativo de igual manera se describirá detalladamente a todo el sistema, para lo cual se hará el uso de diagramas UML. Se utilizará la metodología incremental para la entrega de los módulos del sistema.

3.1 Diseño de modelo del negocio

3.1.1 Modelo del negocio actual

El modelo de negocio es una técnica para diseñar el funcionamiento de una organización a través de distintos diagramas.

Estos serán representados mediante diagramas de flujo detallando todas las tareas requeridas incluyendo a todos los actores involucrados en el proceso.

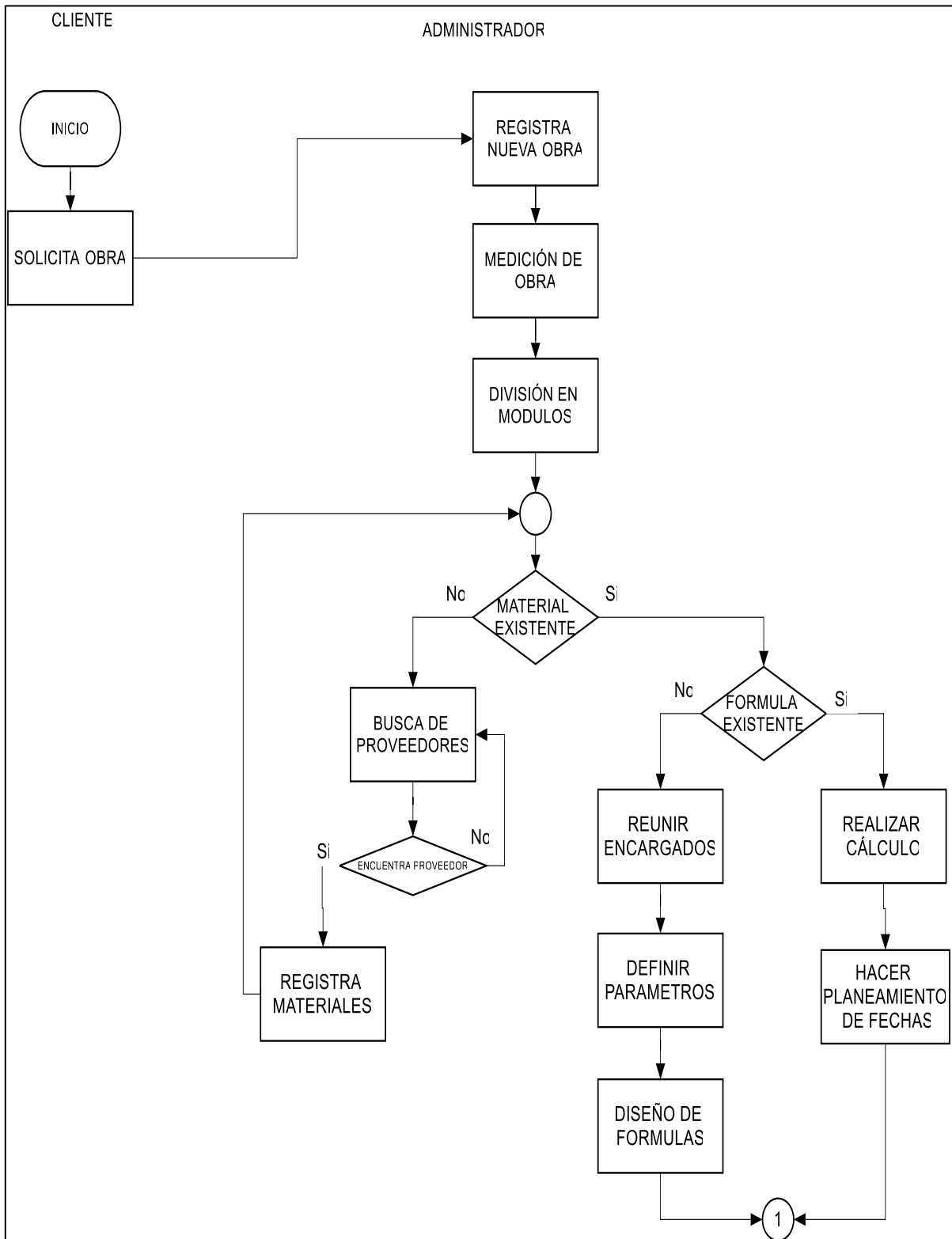
Luego de obtener toda la información necesaria a través de entrevistas realizadas al dueño de la empresa “JBBL”, se realiza los diagramas de negocio de distintos módulos que fueron requeridos por el mismo.

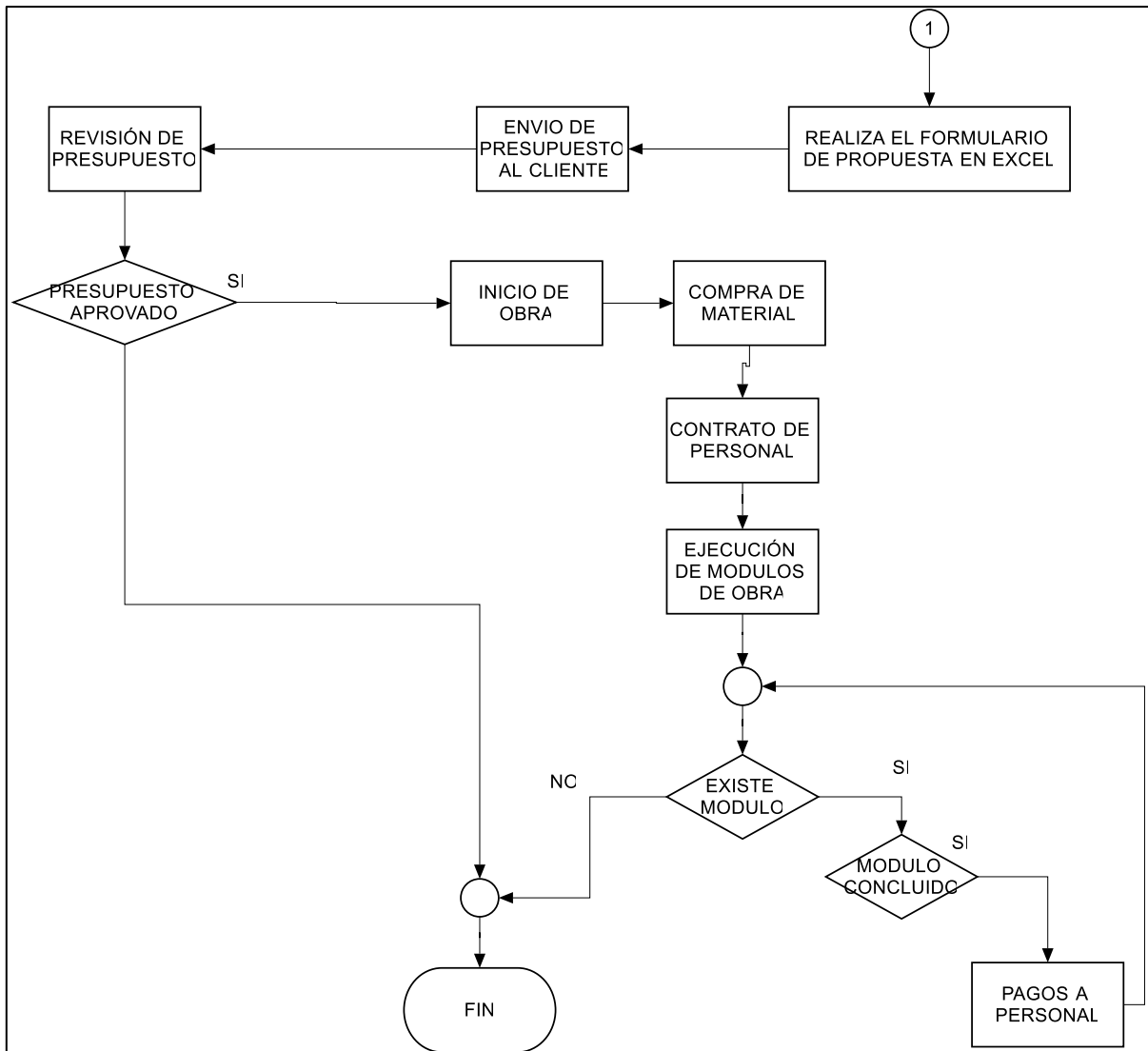
Se diseñaron dos diagramas de negocio, uno actual y el otro alternativo, esto para tener un conocimiento claro del negocio. En el primer diagrama se hace un estudio de cómo funciona la empresa actualmente y la segunda es una alternativa propuesta para solucionar ciertas falencias y mejorar la producción de la empresa.

A continuación, se muestra el modelo de negocio realizado a la empresa.

Figura N° 3. 1

Diagrama de modelo actual





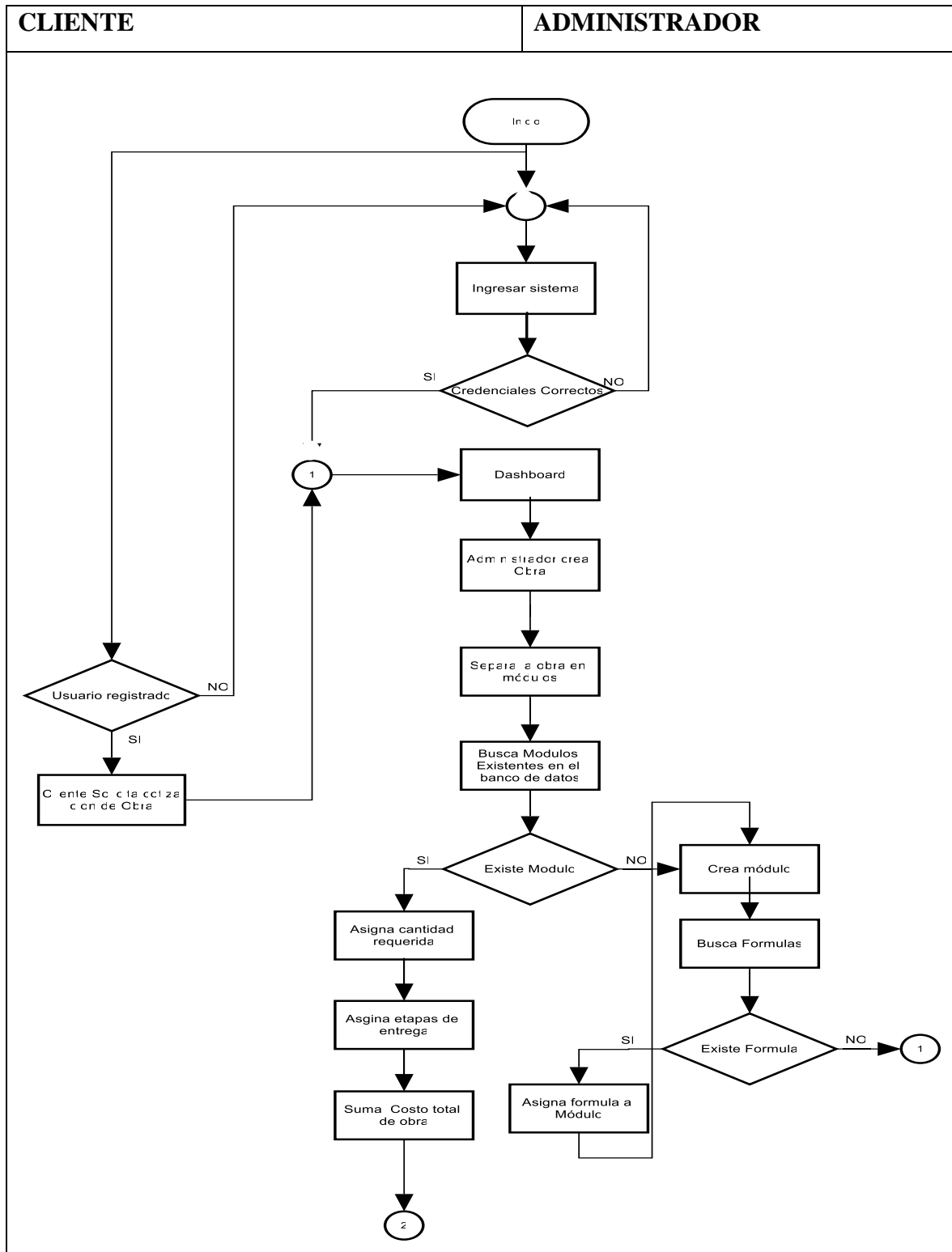
Fuente: [Elaboración propia]

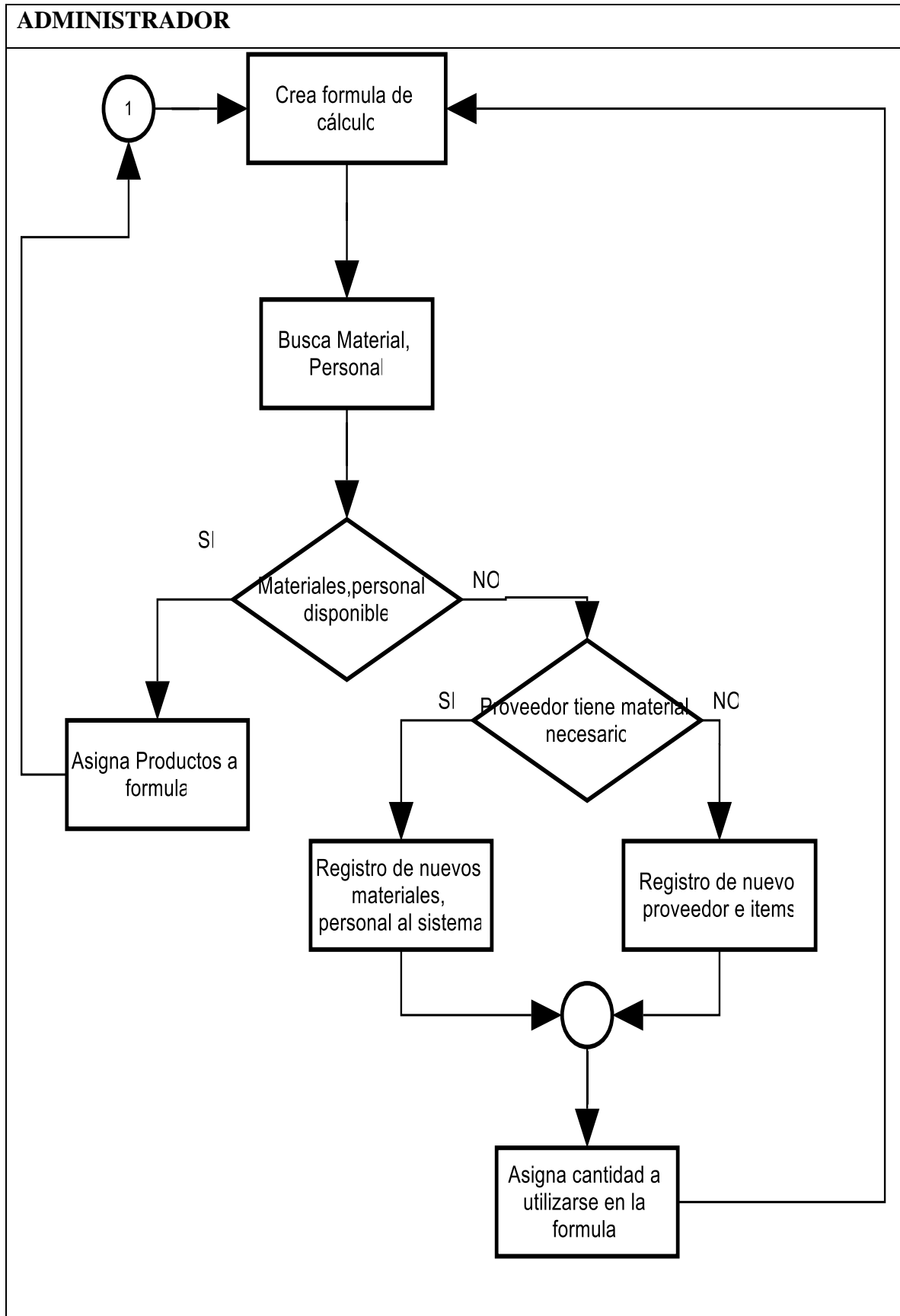
3.1.2 Modelo de negocio alternativo

Con el actual modelado de negocio actual se identificó varias deficiencias lo que conllevó a realizar una propuesta con un nuevo conjunto de procedimientos y soluciones, a continuación, se muestra el flujo de trabajo propuesto.

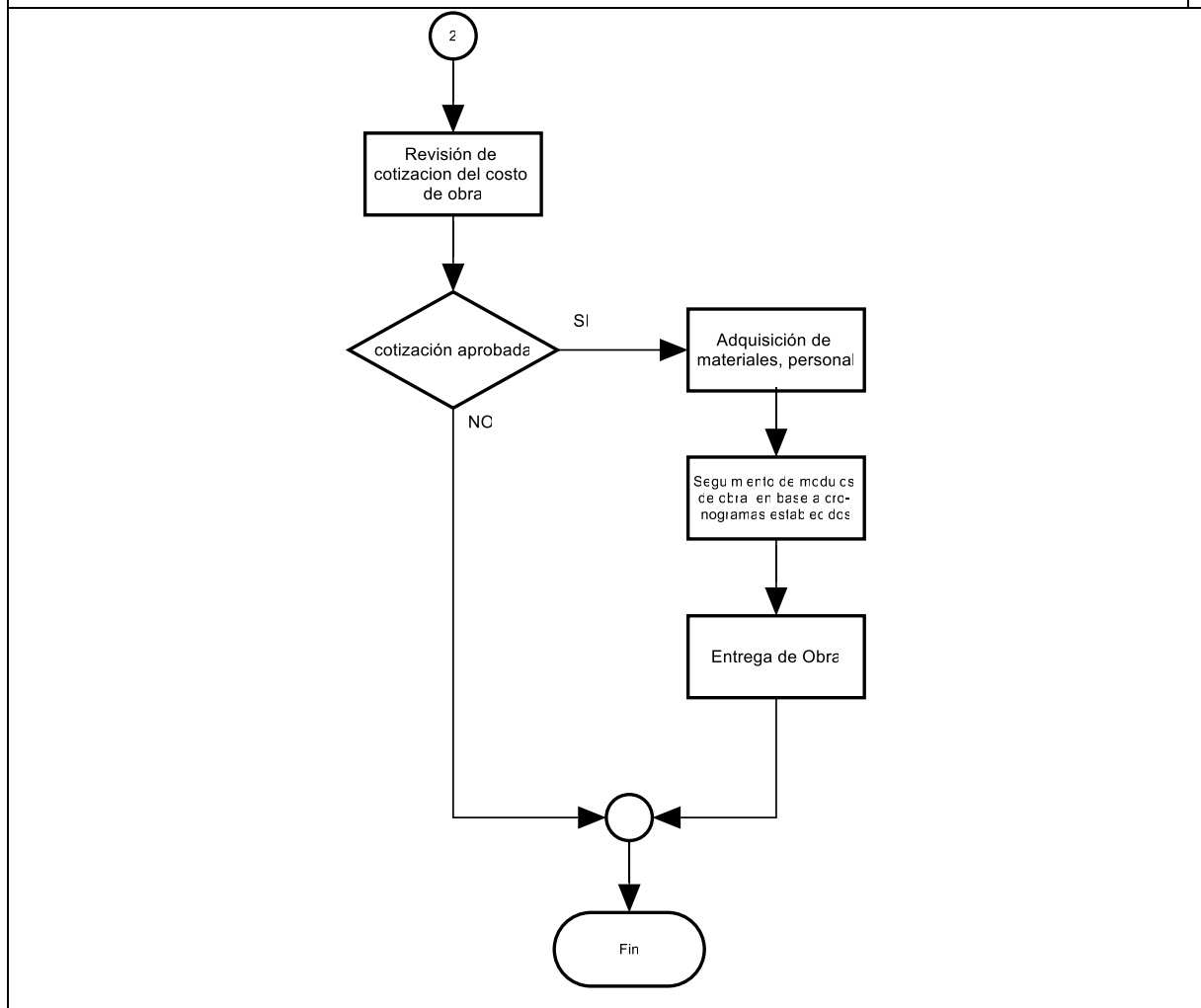
Figura N° 3. 2

Modelo de negocio alternativo





ADMINISTRADOR



Fuente: [Elaboración propia]

3.1.3 Planificación del desarrollo del proyecto

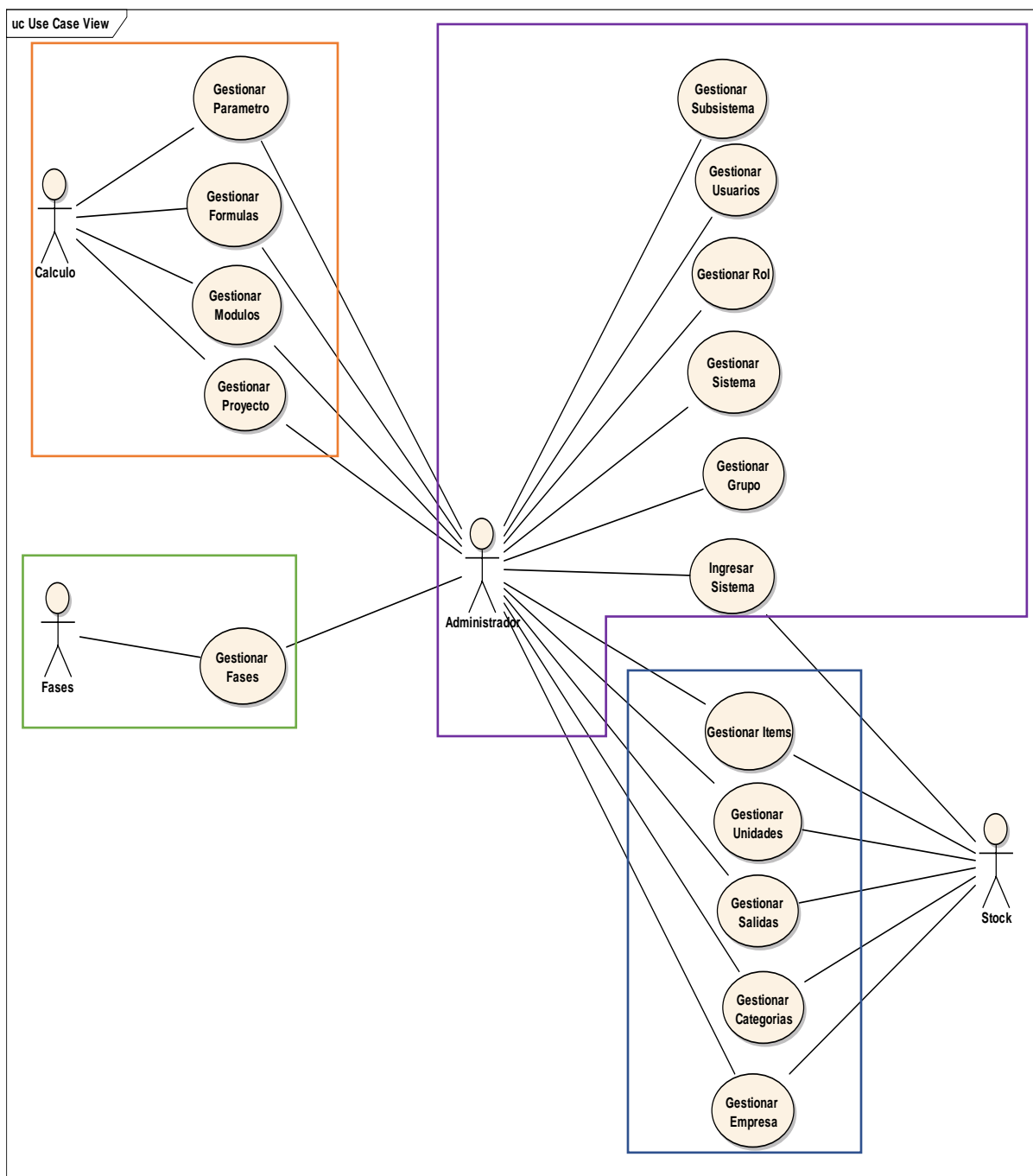
En la elaboración de la planificación del proyecto en el capítulo 2 se justificó la elección del método incremental como la metodología de desarrollo. A continuación, se muestra la planificación de incrementos para el proyecto.

3.1.4 Diseño del caso de uso general del sistema.

Se hizo el diagrama de caso de uso general de todo el sistema, el cual se muestra en la siguiente figura.

Figura N° 3. 3

Diseño de caso de uso general



Fuente: [Elaboración propia]

Por la complejidad del sistema se decidió dividir las entregas en incrementos, a continuación, se describen los mismos.

Tabla N° 3. 1

Planificación de incrementos

N°	Incremento	Descripción
1	Modelar la arquitectura de microservicios para todo el proyecto. Desarrollar el subsistema de autenticación y usuarios.	Se modela la arquitectura que será usada en todo el proyecto y el servidor de autenticación que será reutilizado en todos los subsistemas.
2	Desarrollar el subsistema de gestión de Stock.	Gestiona todos los ítems que serán necesarios en un proyecto.
3	Desarrollar el subsistema de cálculo.	Permite saber cuánto es el costo total del proyecto.
4	Desarrollar el subsistema de fases.	El subsistema permite ver fechas de trabajo del proyecto por modulo.

Fuente: [Elaboración propia]

3.2 Primer Incremento: Modelar la arquitectura de microservicios y desarrollar el subsistema de roles y usuarios

En esta fase se hará todo el diseño de diagramas de clases, diagramas de caso de uso, diagramas de colaboración, modelo y diseño de la base de datos, codificación y pruebas.

3.2.1 Análisis de requerimientos del primer incremento

En el primer incremento se desarrollará el subsistema de autenticación y usuarios que incluirá registro de usuarios, registro de sistemas, roles y grupos para la asignación de roles a usuarios para el acceso a los subsistemas.

Para esto se hizo las entrevistas correspondientes y se llegó a un acuerdo para levantar los requerimientos que este necesita.

A continuación, se lista los requerimientos funcionales y no funcionales.

Requerimientos funcionales

- Gestión de usuarios.
- Gestión de roles.
- Gestión de empresas.
- Gestión de sistemas.
- Gestión de grupos.

Requerimientos no funcionales

- El sistema debe funcionar correctamente en los navegadores web: Chrome, Opera.
- Para el uso del sistema el computador requiere la conexión a internet.

3.2.1.1 Identificación y descripción de actores

Se identifico el actor administrador.

Figura N° 3. 4

Actor Primer Incremento



Fuente: [Elaboración Propia]

Se encargará de gestionar distintos subsistemas crear nuevos proyectos, asignar roles, grupos a distintos usuarios por empresa. Permitirá registrar nuevos subsistemas que se acoplen al

proyecto. Solo este tipo de rol de usuario será capaz de crear nuevos usuarios y asignarlos a los grupos.

3.2.1.2 Identificación de casos de uso por actor

A continuación, se muestra los casos de uso por actor.

Figura N° 3. 5

Diagrama de caso de uso del subsistema de usuario y autenticación.



Fuente: [Elaboración propia]

3.2.1.3 Casos de casos del primer incremento

En las siguientes tablas se describirán los casos de uso del subsistema de autenticación y usuarios.

Caso de Uso: Ingresar al sistema

Permite verificar si el usuario existe y si está registrado en el sistema. Eso dependerá del rol que tenga asignado al sistema.

Tabla N° 3. 2

Caso de uso: Ingresar al sistema

Propósito:	El subsistema de autenticación verifica si el usuario que pide ingresar al sistema está registrado.
Precondiciones:	<ul style="list-style-type: none">• Servidor de usuarios implementado.• Tener un usuario para ingresar.• Conexión a la base de datos.
Postcondiciones:	Redireccionamiento al sistema que solicitó el usuario. Acceso o denegación al sistema.
Circunstancias de uso:	Usado por todos los servicios de la arquitectura.

Fuente: [Elaboración propia]

Caso de Uso: Gestionar sistemas.

Permite registrar nuevos módulos que se acoplarán al sistema mediante registro de url.

Tabla N° 3. 3

Caso de uso: Gestionar sistemas

Propósito:	El usuario podrá registrar los nuevos módulos que se acoplen al sistema.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.

	<ul style="list-style-type: none"> • Tener privilegios para administrar los sistemas.
Postcondiciones:	Almacenamiento en la base de datos de sistemas.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar usuario.

Permite registrar nuevo usuario para el uso del sistema asignando sistemas al mismo para que pueda acceder a ellas.

Tabla N° 3. 4

Caso de uso: Gestionar usuarios

Propósito:	Registra nuevos usuarios, busca, modifica y da de baja.
Precondiciones:	<ul style="list-style-type: none"> • Conexión a la base de datos. • Tener privilegios para administrar los sistemas en el servidor de autenticación de usuarios. • Solo se dará de baja al usuario. • Podrá Editar datos del usuario.
Postcondiciones:	Almacenamiento en la base de datos de sistemas.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de Uso: Gestionar empresa

El administrador del sitio podrá registrar nuevas empresas que usen el sistema asignando un usuario a ella.

Tabla N° 3. 5

Caso de uso: Gestionar empresas

Propósito:	Registra nuevas empresas asignándole un usuario para su uso.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar los sistemas en el servidor de usuarios.
Postcondiciones:	Almacenamiento en la base de datos de sistemas.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de Uso: Gestionar roles

Se podrá crear nuevos roles.

Tabla N° 3. 6

Caso de uso: Gestionar roles

Propósito:	Registro de nuevos roles para diferenciar el acceso al sistema.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar los sistemas en el servidor de usuarios.
Postcondiciones:	Almacenamiento en la base de datos de sistemas.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de Uso: Gestionar grupos

En un grupo se podrá ingresar el rol, sistema y empresa que luego será asignado a un usuario

Tabla N° 3. 7

Caso de uso: Gestionar grupos

Propósito:	Registra grupos de roles a cada usuario para que estos puedan tener acceso o no al sistema.
Precondiciones:	<ul style="list-style-type: none"> • Conexión a la base de datos. • Tener privilegios para administrar los sistemas en el servidor de usuarios.
Postcondiciones:	Almacenamiento en la base de datos de sistemas.
Circunstancias de uso:	Restringido solo para administradores del sistema.

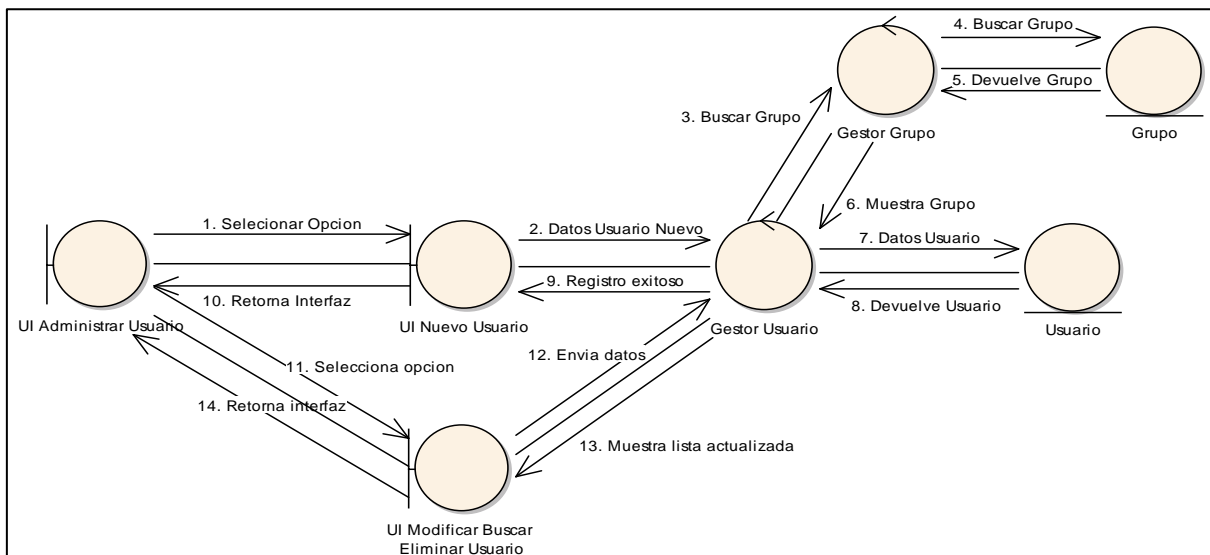
Fuente: [Elaboración propia]

3.2.1.4 Diagramas de colaboración

A continuación, se mostrarán en las figuras los diagramas de colaboración, entre los cuales se observa las figuras existentes.

Figura N° 3. 6

Diagrama de colaboración: Gestionar usuario

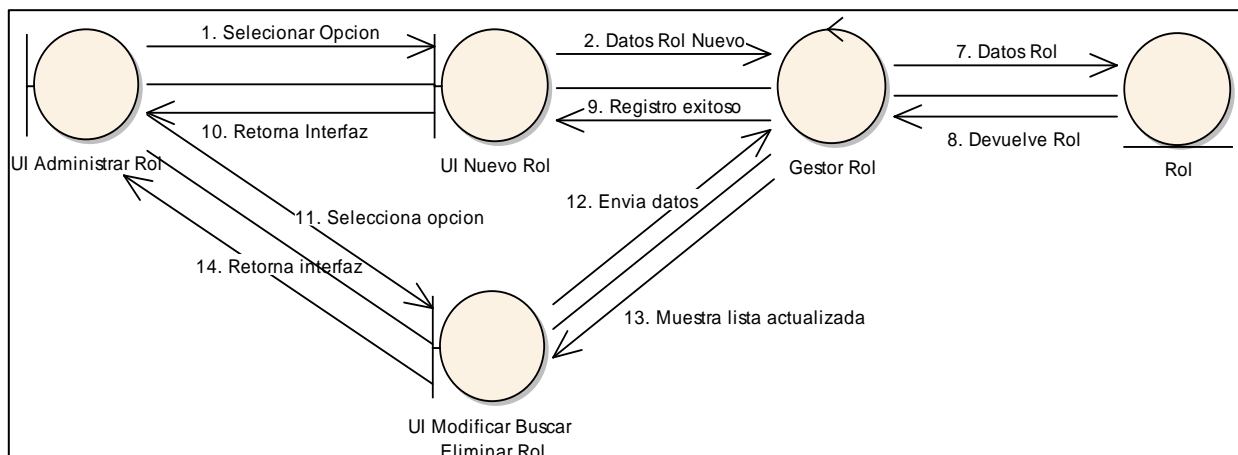


Fuente: [Elaboración propia]

El administrador puede crear distintos usuarios con distintos grupos y dar permisos a cada sistema, los datos serán almacenados en la tabla de usuarios del subsistema de usuarios.

Figura N° 3. 7

Diagrama de colaboración: Gestionar rol

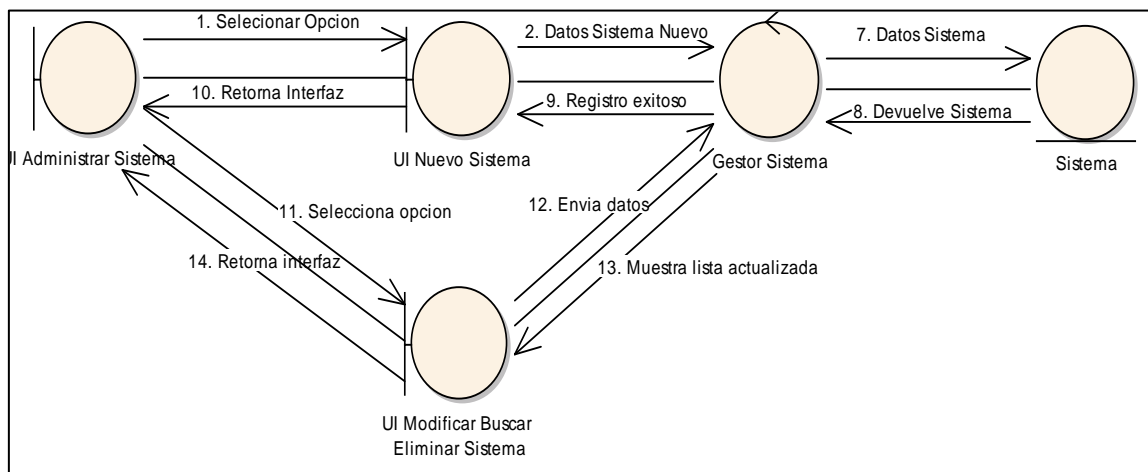


Fuente: [Elaboración propia]

Los roles serán asignados a un usuario por el administrador, los registros serán almacenados en la tabla de roles.

Figura N° 3. 8

Diagrama de colaboración: Gestionar sistema

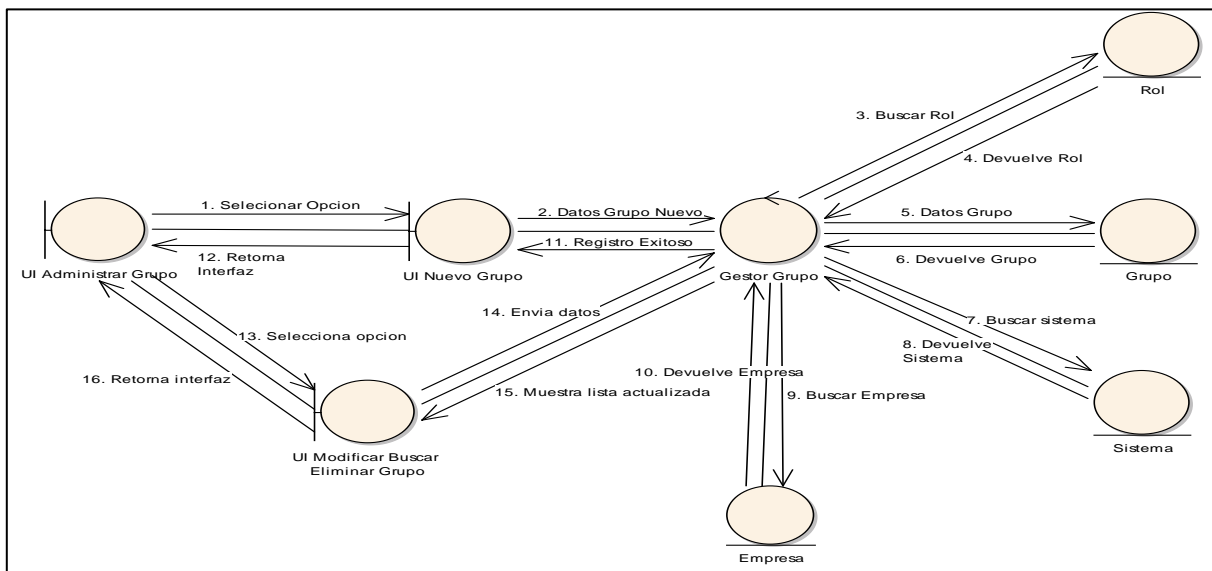


Fuente: [Elaboración propia]

Estos serán gestionados cuando haya nuevos módulos o subsistemas en el sistema, a estas solo tendrá acceso el administrador, serán almacenados en la tabla sistemas del subsistema de usuarios.

Figura N° 3. 9

Diagrama de colaboración: Gestionar grupo

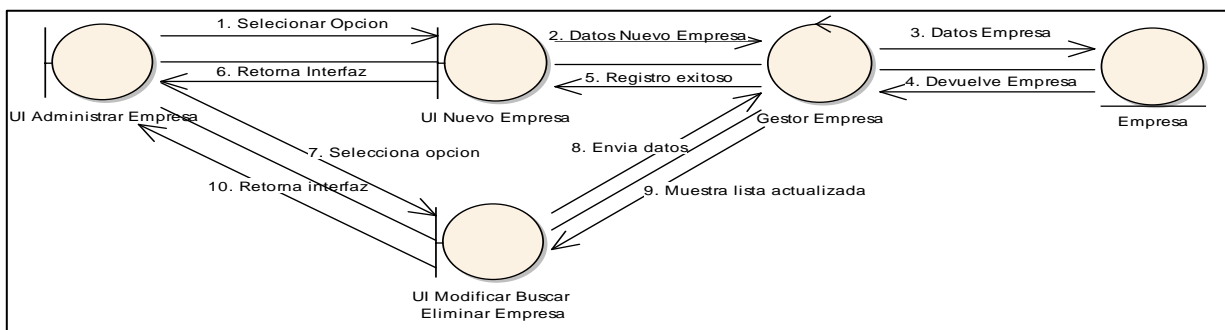


Fuente: [Elaboración propia]

Los grupos son creados y almacenados en la tabla de grupos y son usadas para asignar acceso a usuario a ciertos subsistemas con roles distintos.

Figura N° 3. 10

Diagrama de colaboración: Gestionar Empresa



Fuente: [Elaboración propia]

Las empresas serán registradas y almacenadas en la tabla de empresas en el subsistema de usuarios.

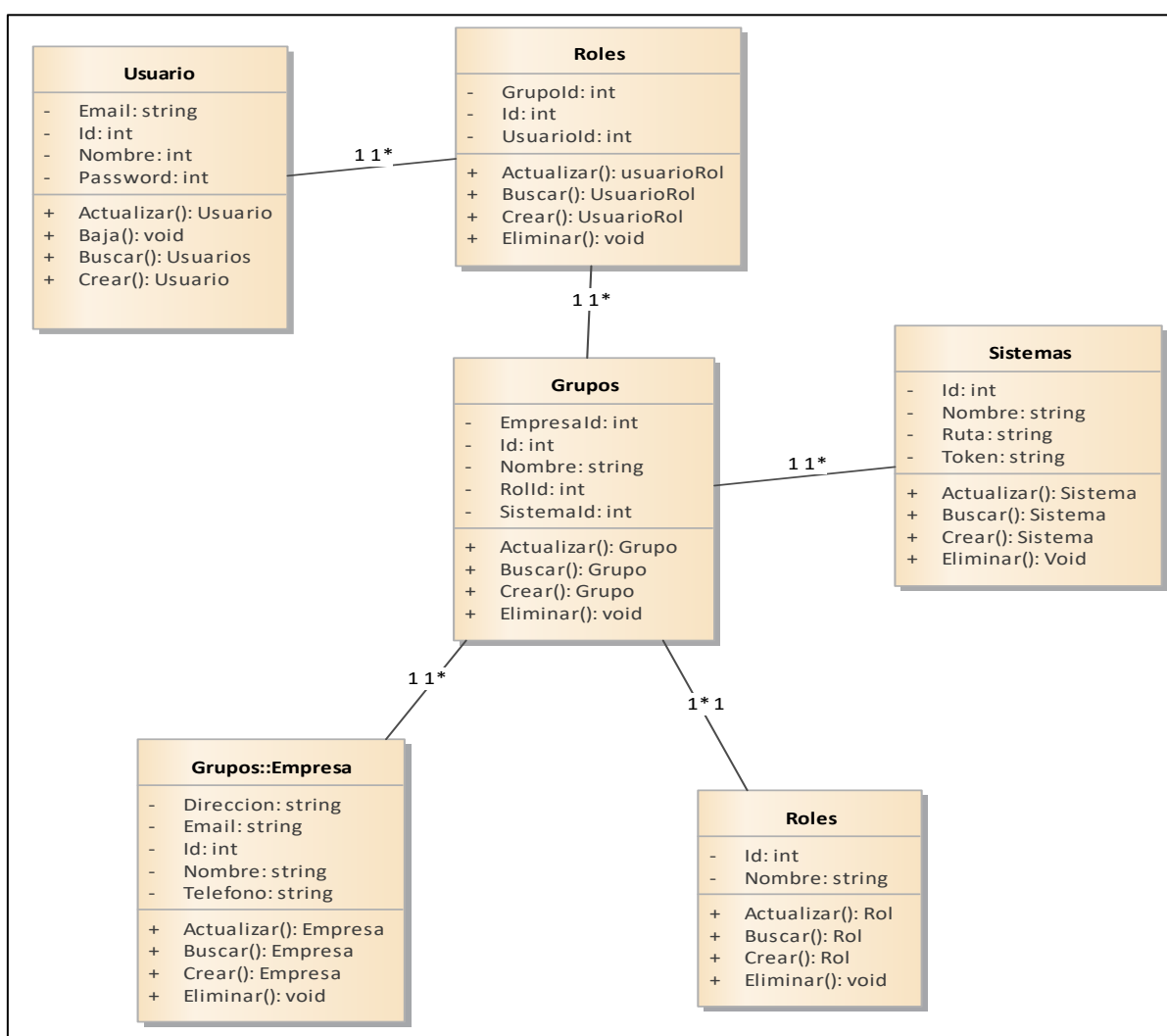
3.2.2 Diseño del primer incremento

3.2.2.1 Diagrama de clases

En la siguiente figura se muestra el diagrama de clases, con las relaciones que existen entre ellas.

Figura N° 3. 11

Diagrama de clases del primer incremento

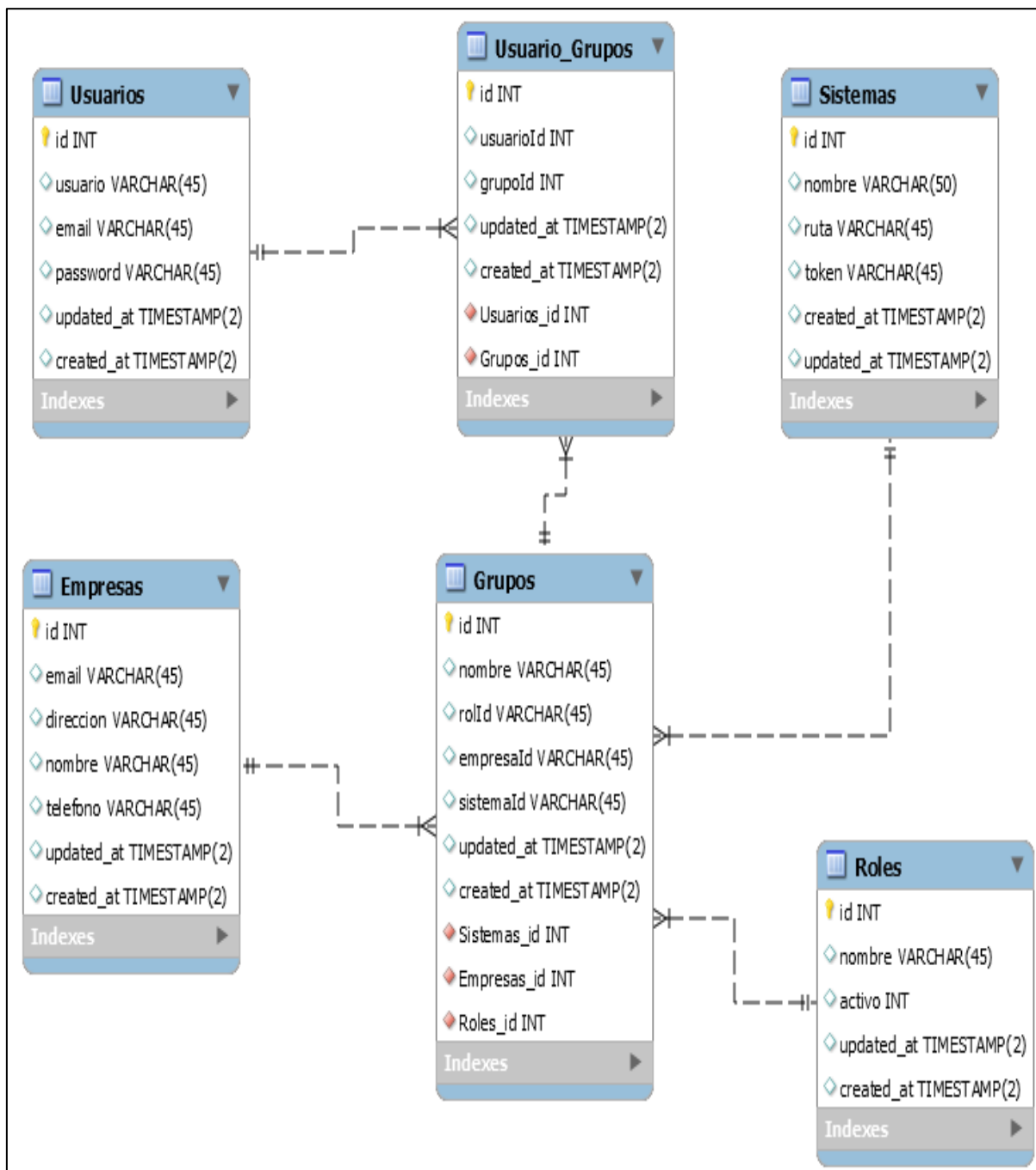


Fuente: [Elaboración propia]

3.2.2.2 Diseño de la base de datos

Figura N° 3. 12

Base de datos del primer incremento



Fuente: [Elaboración propia]

3.2.2.3 Diccionario de datos

En las siguientes tablas se detallará las tablas usadas en el sistema de usuario.

Tabla N° 3. 8

Tabla Usuarios

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra id único de usuario.
nombre	Varchar	No	No	No	Registra Nombre del usuario.
email	Varchar	No	No	No	Registra el email del usuario.
contrasena	Varchar	No	No	No	Se almacena la contraseña encriptada del usuario en la base de datos.

Fuente: [Elaboración propia]

Tabla N° 3. 9

Diccionario de datos de la tabla Empresas

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único de la empresa.
nombre	Varchar	No	No	No	Registra el nombre de la empresa.
direccion	Varchar	Si	No	No	Registra la ubicación de la empresa.
email	Varchar	Si	No	No	Registra el email institucional de la empresa.
telefono	Varchar	Si	No	No	Número telefónico de referencia.

Fuente: [Elaboración propia]

Tabla N° 3. 10

Diccionario de datos de la tabla Grupos

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra código único del grupo.
nombre	Varchar	No	No	No	Registra nombre de la empresa.
rol_id	Int	No	No	Si	Registra la clave primaria de rol.
empresa_id	Int	No	No	Si	Registra la clave primaria de empresas.
sistema_id	Int	No	No	Si	Registra la clave primaria de sistemas.

Fuente: [Elaboración propia]

Tabla N° 3. 11

Diccionario de datos de la tabla Sistemas

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra código único de Sistema.
nombre	Varchar	No	No	No	Registra nombre del sistema.
ruta	Varchar	No	No	No	Registra la ruta del subsistema.
token	Varchar	No	No	No	Registra el token de la empresa.

Fuente: [Elaboración propia]

Tabla N° 3. 12

Diccionario de datos de la tabla Roles

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra código único de rol.

nombre	Varchar	No	No	No	Registra el nombre
--------	---------	----	----	----	--------------------

Fuente: [Elaboración propia]

Tabla N° 3. 13

Diccionario de datos de la tabla Usuario_Grupos

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra código único de rol.
grupoId	Int	No	No	Si	Registra clave primaria del grupo.
usuarioId	Int	No	No	Si	Registra clave primaria del usuario.

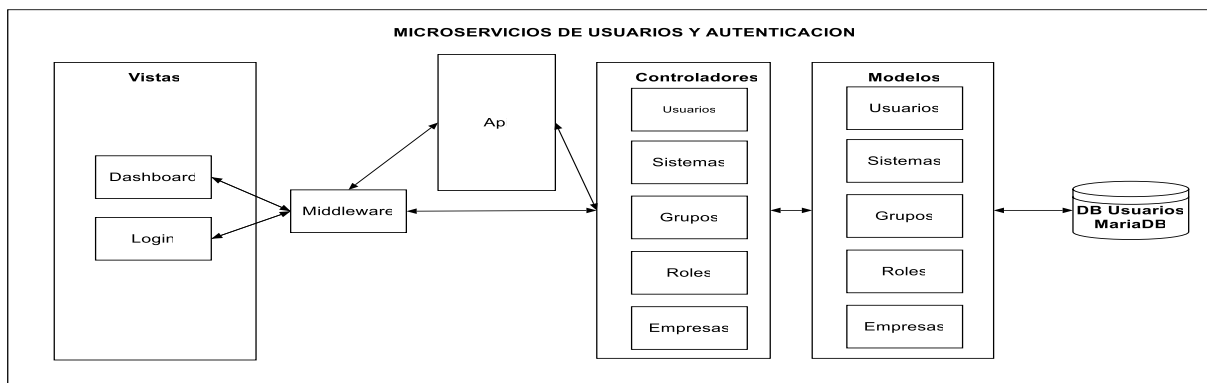
Fuente: [Elaboración propia]

3.2.2.4 Diseño de la arquitectura del software

La siguiente imagen muestra la arquitectura del sistema Web basado en microservicios.

Figura N° 3. 13

Arquitectura del subsistema de usuarios



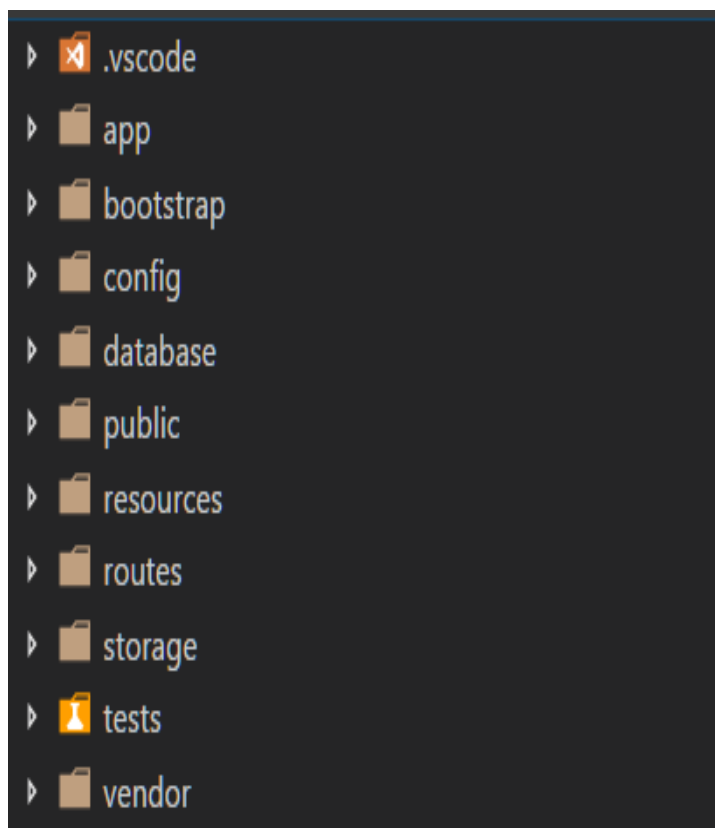
Fuente: [Elaboración Propia]

3.2.3 Codificación del primer incremento

El proyecto se organiza en una estructura de archivos, que son categorizados de acuerdo con la necesidad de uso y que también es propuesto por el framework Laravel.

Figura N° 3. 14

Estructura de archivos del subsistema de Usuarios



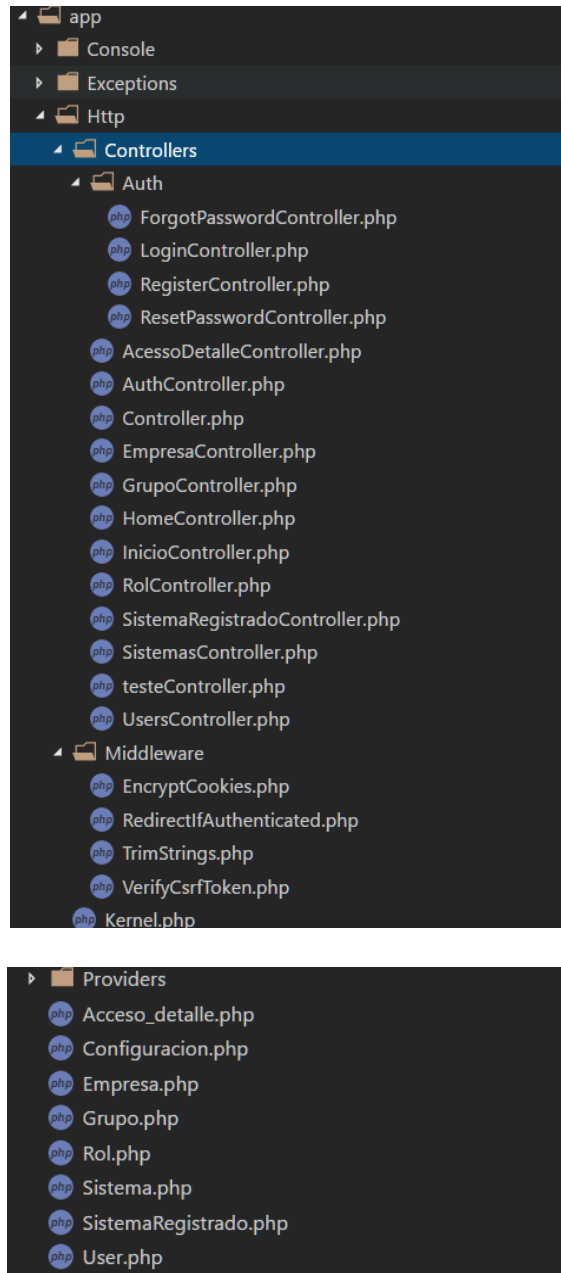
Fuente: [Elaboración propia]

Para trabajar en el proyecto es necesario el uso de herramientas que nos provee el framework Laravel, las que son controladores, middlewares, interfaces, routing, modelos, mapeadores de datos, librerías de terceros, servicios.

En la carpeta app se encuentra los controladores, los modelos, los middleware y configuraciones para el backend del proyecto.

Figura N° 3. 15

Estructura de archivos de controladores modelo y middleware de Usuarios

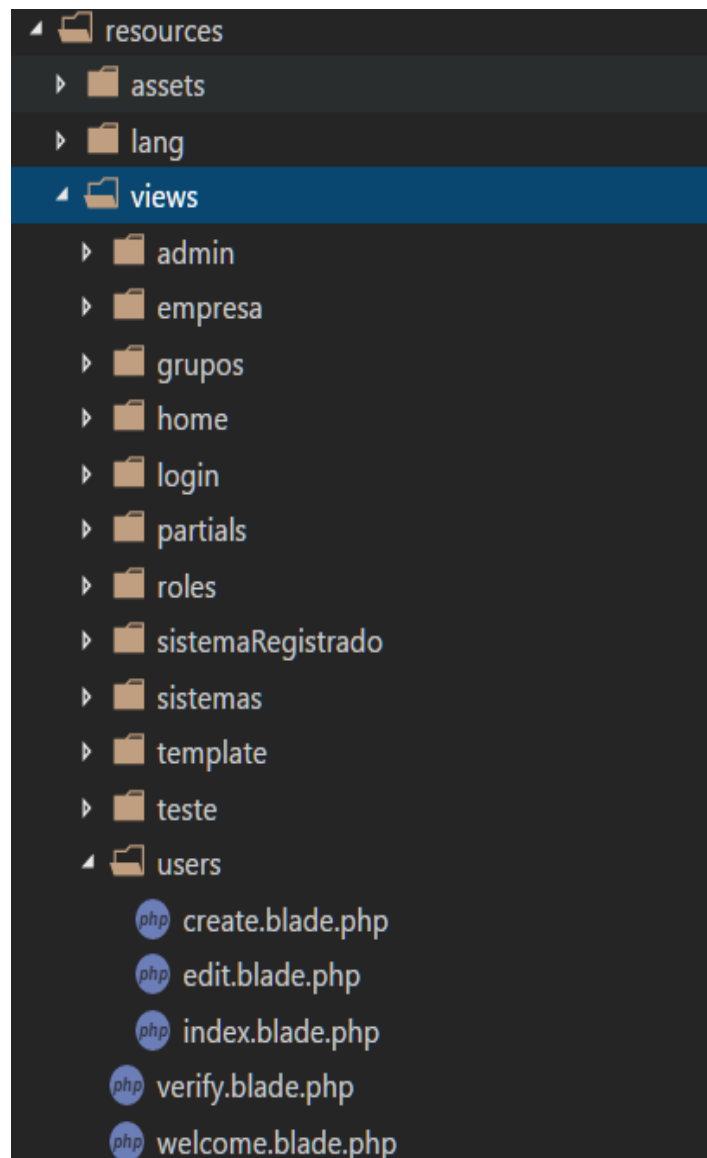


Fuente: [Elaboración Propia]

Para la interacción del usuario es necesario las vistas, estas se encuentran en la carpeta de resources.

Figura N° 3. 16

Organización de las vistas para el primer incremento



Fuente: [Elaboración propia]

3.2.3.1 Implementación y funcionalidades

En este apartado se mostrará las interfaces del primer incremento. Comenzando con la vista del ingreso al sistema

Figura N° 3. 17

Interfaz de ingreso al sistema

Ingrese sus datos para iniciar session

ronaldwin79@hotmail.com

.....

Ingresar

[Registrate](#)

Fuente: [Elaboración propia]

Una vez ingresado al sistema con las credenciales correctas el sistema le muestra un panel de control en el cual tendrá acceso de acuerdo a los privilegios con los que fue creado.

Figura N° 3. 18

Panel de control del subsistema de Usuarios

U Stock Calculo Usuarios Salir

Usuarios

Show 10 entries Search:

Nombre	Email		
rluna	rluna@luna.com	Edit	Delete
ronald	ronaldwin79@hotmail.com	Edit	Delete
test	test@mail.com	Edit	Delete

Showing 1 to 8 of 8 entries

Previous 1 Next

+ Agregar

Fuente: [Elaboración propia]

Figura N° 3. 19

Formulario de registro de usuarios

Editar

Nombre

ronald

Email

ronaldwin79@hotmail.com

Password

Asignar Grupos

✕ Usuario Stock

✕ JBBL Admin

Cancelar

Guardar

Fuente: [Elaboración propia]

Figura N° 3. 20

Panel de control de grupos

U

Stock

Calculo

Usuarios

Salir

Grupos

Show 10 entries

Search:

Nombre	Rol	Sistema	
JBBL Admin	Almacen los Olivos	admin	<div>EditDelete</div>
Usuario Stock	JBBL	admin	<div>EditDelete</div>

Showing 1 to 2 of 2 entries

Previous

1

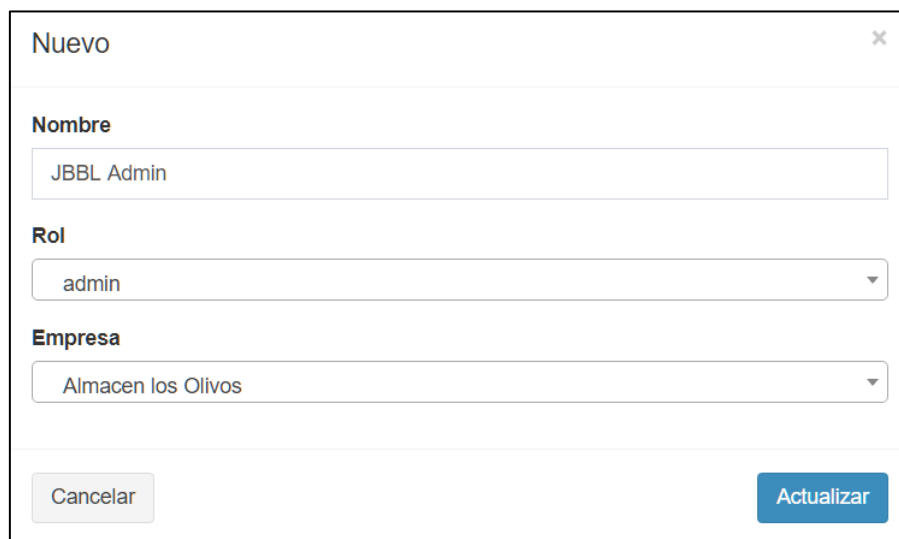
Next

+ Agregar

Fuente: [Elaboración propia]

Figura N° 3. 21

Formulario registro de grupos



Nuevo

Nombre

JBBL Admin

Rol

admin

Empresa

Almacen los Olivos

Cancelar Actualizar

Fuente: [Elaboración propia]

3.2.4 Casos de prueba del primer incremento

Se hicieron pruebas de uso en el sistema del primer incremento en el subsistema de usuarios.

Tabla N° 3. 14

Casos de prueba para la verificación de funcionalidad de ingreso al sistema

Caso de prueba	Descripción	Prueba Realizada	Resultado Esperado
	Con una cuenta asignada el usuario	Una vez ingresada los datos correctos del usuario, el sistema le redireccionara al subsistema que solicito al iniciar sesión.	Redirecciona a la página principal que solicito el usuario. [Correcto]
		Cuando ingrese usuario o contraseña	Muestra mensaje de validación indicando

Ingresar al sistema.	podrá acceder al sistema.	incorrecta el sistema mostrara un mensaje indicando que no tiene las credenciales correctas.	que las credenciales son incorrectas. [Correcto]
		Deberá ser obligatorio llenar los campos requeridos para ingresar al sistema.	El sistema muestra un mensaje de validación indicando que debe llenar los campos. [Correcto]
Registro de nuevo usuario.	Se deberá registrar nuevos usuarios.	Al guardar se debe verificar que los datos no sean duplicados.	En caso de datos existentes el sistema muestra un mensaje de error. [Correcto]
		Debe verificarse que los registros que se guarden no estén en vacíos.	Muestra el mensaje de validación indicando que los campos no pueden ir en blanco o nulos. [Correcto]
		Deberá poder asignarse uno o más grupos de acceso a los subsistemas a cada usuario.	Se verifico que tiene un control para seleccionar diversos grupos y son asignados al usuario.

			[Correcto]
Actualizar datos de usuario.	Se deberá poder modificar los datos de usuario.	Deberá traer los datos correctos del usuario para modificarlos.	El sistema devuelve correctamente los datos solicitados. [Correcto]
		Modificando los datos del usuario el sistema deberá validar si los campos están vacíos.	Muestra un mensaje de validación indicando que debe llenar los campos necesarios. [Correcto]
		En el campo de grupos de acceso modificará a los sistemas que tendrá acceso el usuario.	Es posible asignar uno o más grupos de acceso a cada usuario. [Correcto]
Buscar usuarios.	Se deberá poder buscar usuarios.	El sistema cuenta con un textbox de búsqueda al cual ingresando datos del usuario buscara el registro requerido.	El buscador es en tiempo real y entrega los datos requeridos y los más aproximados al solicitado. [Correcto]

Fuente: [Elaboración propia]

3.3 Segundo Incremento: Desarrollar el subsistema de stock de materiales

3.3.1 Análisis de requerimientos del segundo incremento

Requerimientos funcionales

- El Administrador podrá registrar, modificar y eliminar datos de unidades.
- El Administrador podrá registrar, modificar y eliminar datos de categorías.
- El Administrador podrá registrar, modificar y eliminar datos de salidas.
- El Administrador podrá registrar, modificar y eliminar datos de ítems.

Requerimientos no funcionales

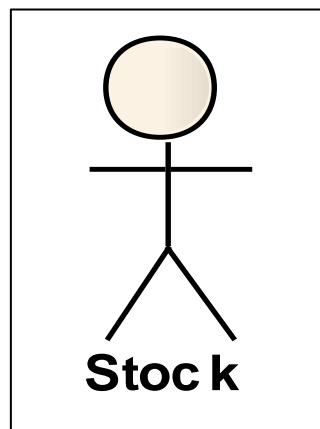
- Uso fácil e intuitivo.
- El sistema debe rechazar cualquier ingreso no autorizado al sistema.

3.3.1.1 Identificación y descripción de actores

Se identifico el actor de stock el cual será representado en los diagramas como se muestra en la siguiente figura.

Figura N° 3. 22

Actor segundo incremento



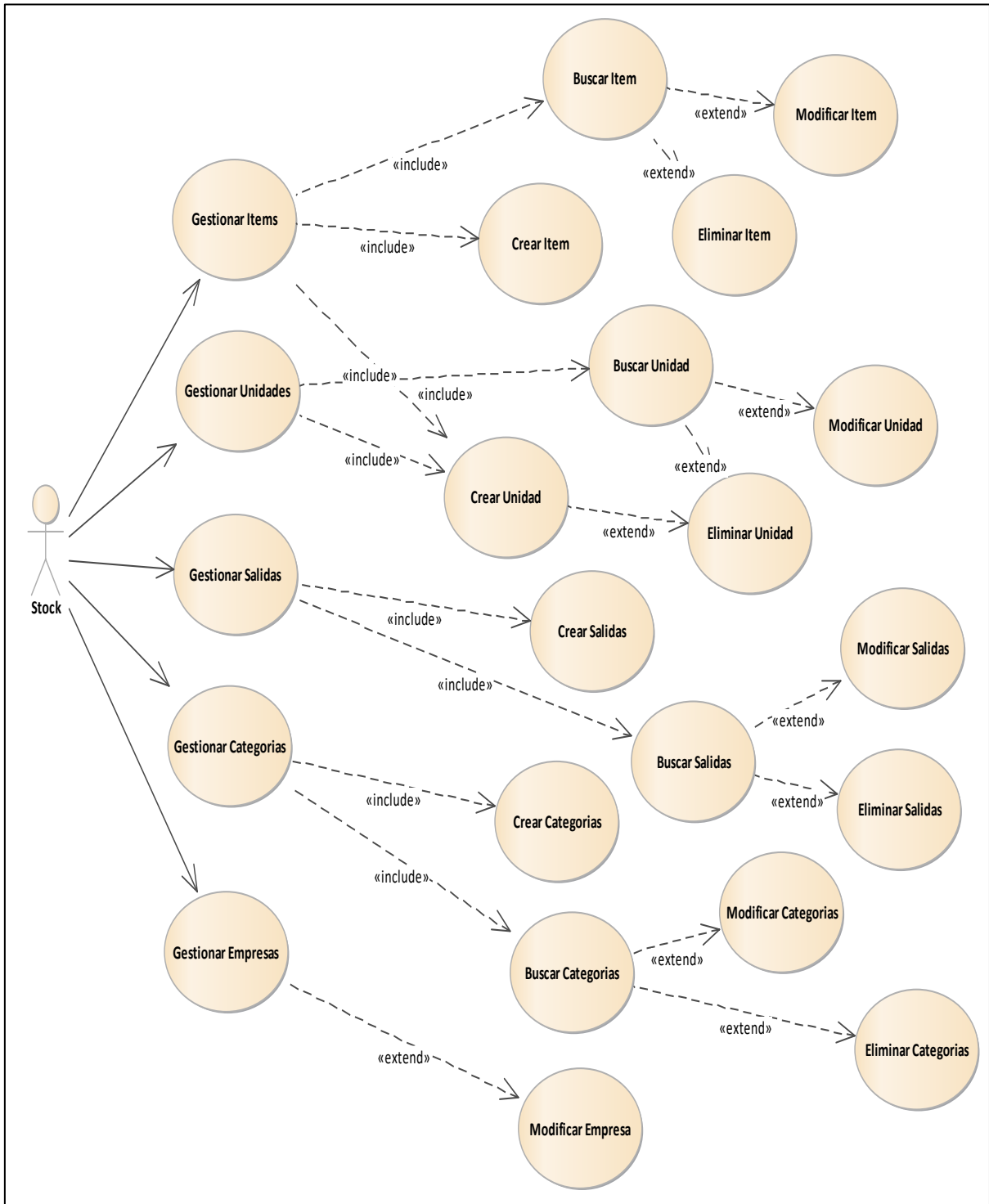
Fuente: [Elaboración propia]

3.3.1.2 Identificación de casos de uso por actor

A continuación, se muestra los casos de uso por actor

Figura N° 3. 23

Diagrama de caso de uso del subsistema de stock



Fuente: [Elaboración propia]

3.3.1.3 Casos de uso del segundo incremento

Caso de uso: Gestionar Unidades

Permite registrar nuevos registros del tipo de unidad que manejara el stock de ítems, estos pueden ser: unidades, metros cúbicos, metro lineal, etc.

Tabla N° 3. 15

Caso de uso: Gestionar unidades

Propósito:	Registro de nuevas unidades para su posterior uso.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.
Postcondiciones:	Almacenamiento en la base de datos de stock.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar categoría.

Usado para categorizar los ítems y poder seleccionar de acuerdo con su uso, estas pueden ser: materiales eléctricos, herramientas, etc.

Tabla N° 3. 16

Caso de uso: Gestionar categorías

Propósito:	Registro de nuevas categorías para agrupar los ítems.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.
Postcondiciones:	Almacenamiento en la base de datos de stock.
Circunstancias de uso:	Restringido solo para los administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar ítems

Permite registrar nuevos ítems que serán usados como insumos en el proyecto que requiera usar.

Tabla N° 3. 17**Caso de uso:** Gestionar ítems

Propósito:	Registro de nuevos ítems.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.• Registro de unidades.• Registro de categorías.
Postcondiciones:	Almacenamiento en la base de datos de stock.
Circunstancias de uso:	Restringido solo para administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar salidas

Permite registrar las salidas de los ítems requeridos, estos pueden ser registrados por el administrador o el subsistema de cálculos.

Tabla N° 3. 18**Caso de uso:** Gestionar salidas

Propósito:	Registro de salidas de ítems.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Ítems registrados para la salida.• Tener privilegios para administrar el sistema.
Postcondiciones:	Almacenamiento en la base de datos de Stock.

Circunstancias de uso:	Restringido solo para administradores del sistema.
-------------------------------	--

Fuente: [Elaboración propia]

Caso de uso: Gestionar empresa

El actor solo podrá actualizar los datos de la empresa registrada ya que los demás permisos están en el subsistema de usuarios y está limitado por el actor administrador.

Tabla N° 3. 19

Caso de uso: Gestionar empresa

Propósito:	Actualización de datos de la empresa.
Precondiciones:	<ul style="list-style-type: none"> • Conexión a la base de datos. • Tener privilegios para administrar el sistema.
Postcondiciones:	Registro en la base de datos en base a salida de ítems.
Circunstancias de uso:	Restringido solo para administradores del sistema.

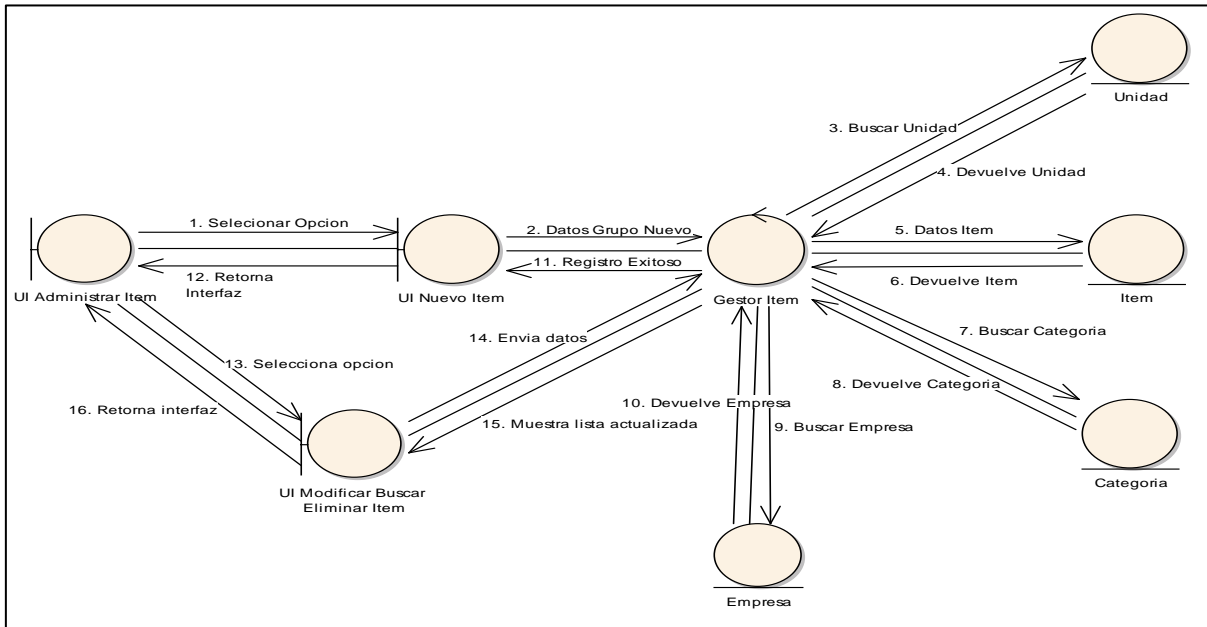
Fuente: [Elaboración propia]

3.3.1.4 Diagramas de colaboración de Stock

En las siguientes figuras se mostrarán los diagramas de colaboración realizados para el subsistema de stock.

Figura N° 3. 24

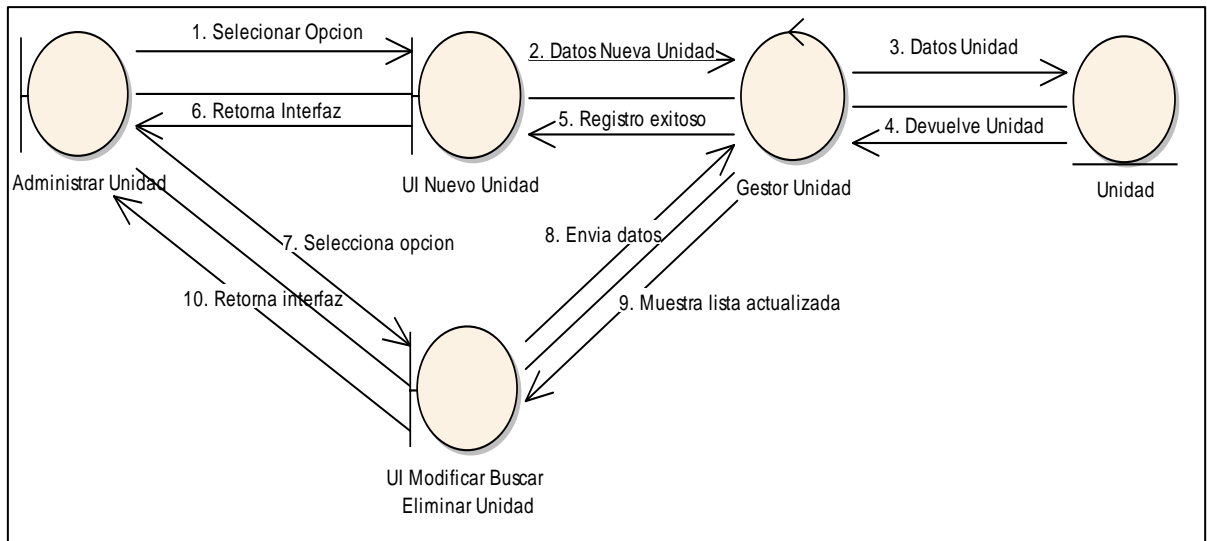
Diagrama de colaboración: Gestionar Ítems



Fuente: [Elaboración propia]

Figura N° 3. 25

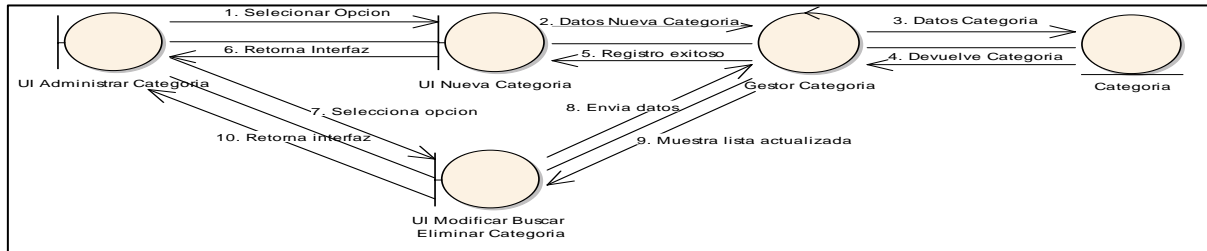
Diagrama de colaboración: Gestionar Unidad



Fuente: [Elaboración propia]

Figura N° 3. 26

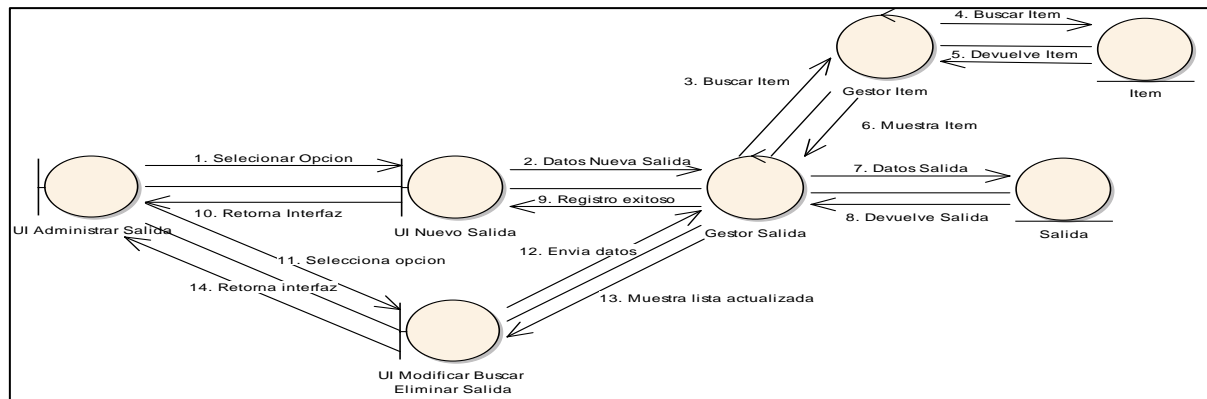
Diagrama de colaboración: Gestionar Categoría



Fuente: [Elaboración propia]

Figura N° 3. 27

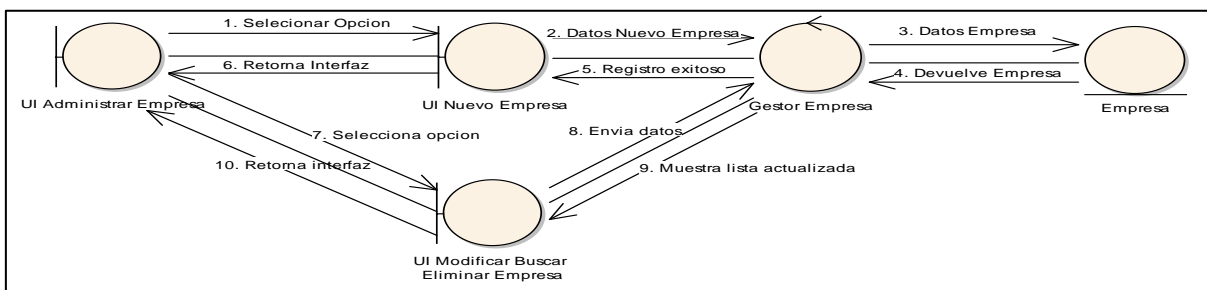
Diagrama de colaboración: Gestionar Salidas



Fuente: [Elaboración propia]

Figura N° 3. 28

Diagramas de colaboración: Gestionar Empresa



Fuente: [Elaboración propia]

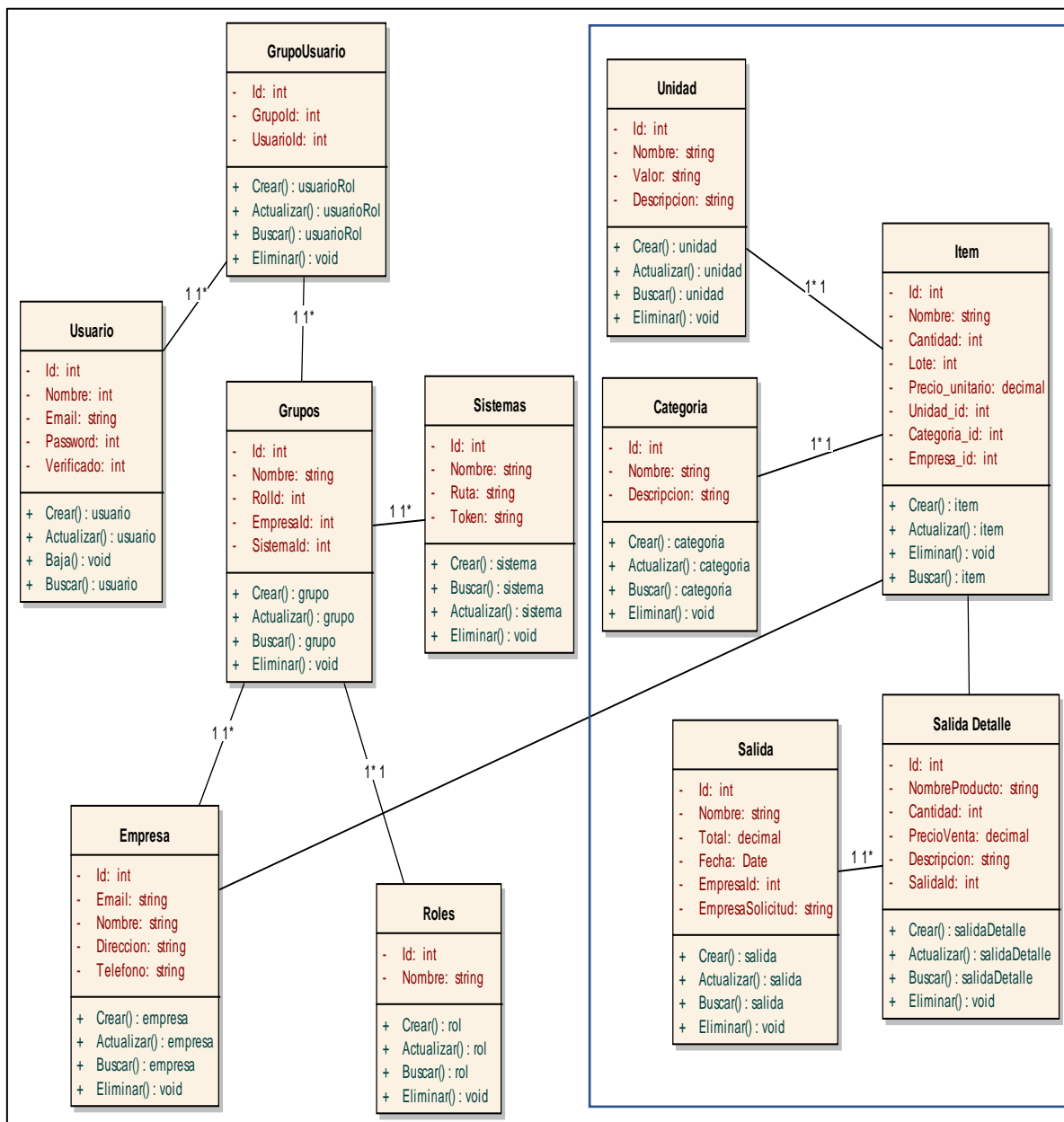
3.3.2 Diseño del segundo incremento

3.3.2.1 Diagrama de clases

Se agrego el diagrama de clases de stock para el segundo incremento.

Figura N° 3. 29

Diagrama de clases para el segundo incremento

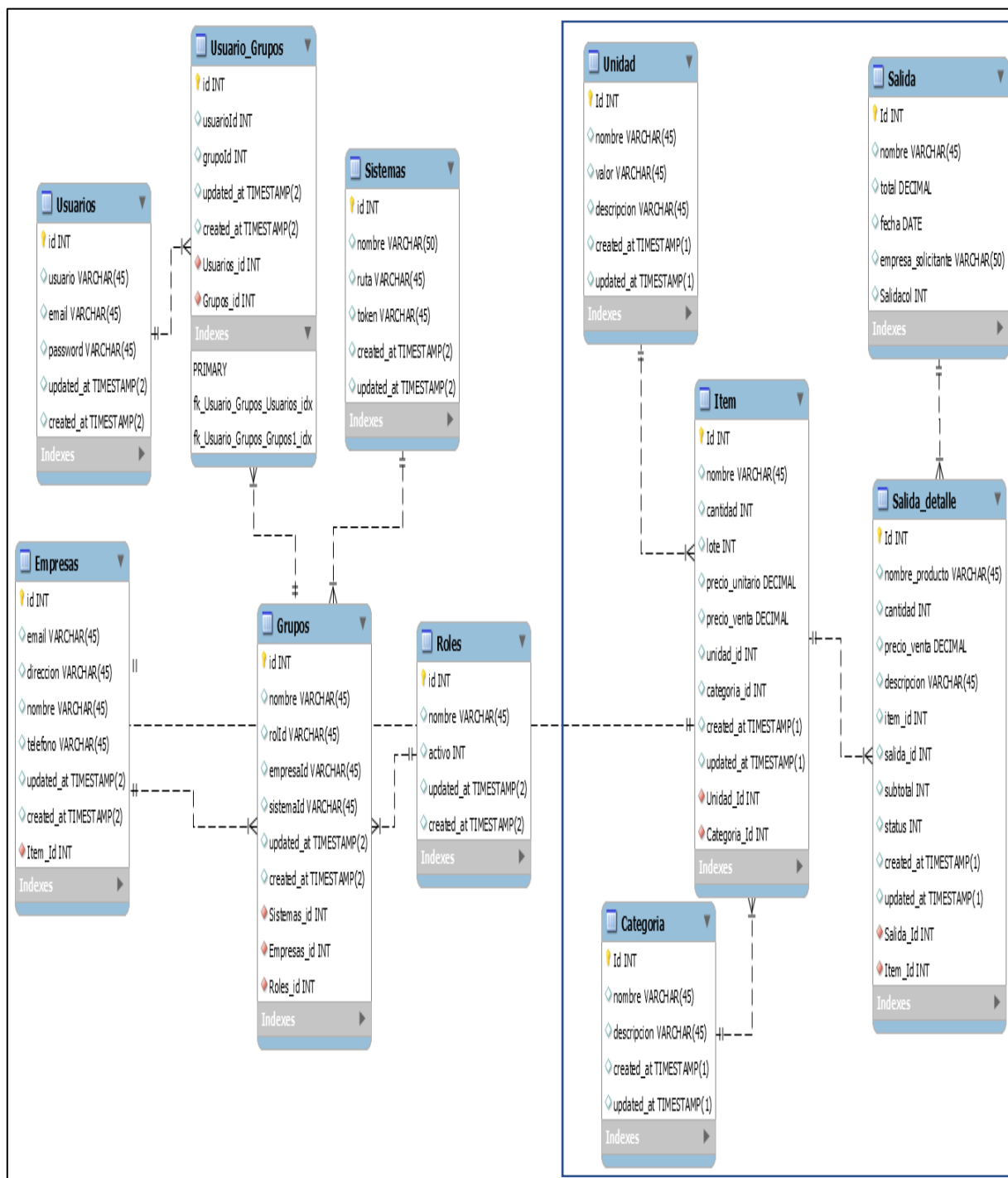


Fuente: [Elaboración propia]

3.3.2.2. Diseño de base de datos

Figura N° 3. 30

Base de datos para el segundo incremento



Fuente: [Elaboración propia]

3.3.2.2 Diccionario de datos

Tabla N° 3. 20

Diccionario de datos de la tabla Unidad

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único de la unidad.
nombre	Varchar	No	No	No	Registra el nombre de la empresa.
descripción	Varchar	No	No	No	Registra la descripción de la unidad.
valor	Varchar	No	No	No	Valor de la unidad.

Fuente: [Elaboración propia]

Tabla N° 3. 21

Diccionario de datos de la tabla Categoría

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único de la categoría.

nombre	Varchar	No	No	No	Registra el nombre de la categoría.
descripción	Varchar	No	No	No	Registra la descripción de la categoría.

Fuente: [Elaboración propia]

Tabla N° 3. 22

Diccionario de datos de la tabla ítem

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único del ítem.
nombre	Varchar	No	No	No	Registra el nombre del ítem.
cantidad	Int	No	No	No	Cantidad de ítem existente.
lote	Int	No	No	No	Cantidad de artículos por ítem.
precio_unitario	Decimal	No	No	No	Precio unitario del ítem.
unidad_id	Int	No	No	Si	Código primario de unidad.

categoria_id	Int	No	No	Si	Código primario de categoría.
--------------	-----	----	----	----	-------------------------------

Fuente: [Elaboración propia]

Tabla N° 3. 23

Diccionario de datos de la tabla Salida_detalle

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único del ítem.
nombre_producto	Varchar	Si	No	No	Nombre del ítem para salida
precio_venta	Decimal	No	No	No	Precio unitario del ítem.
cantidad	Int	No	No	No	Cantidad salida del ítem.
descripción	Varchar	Si	No	No	Detalle del ítem.
subtotal	Decimal	No	No	No	Precio parcial.
status	Int	Si	No	No	Verifica salida.

item_id	Int	No	No	Si	Código primario de ítem.
salida_id	Int	No	No	Si	Código primario de salida.

Fuente: [Elaboración propia]

Tabla N° 3. 24

Diccionario de datos de la tabla Salida

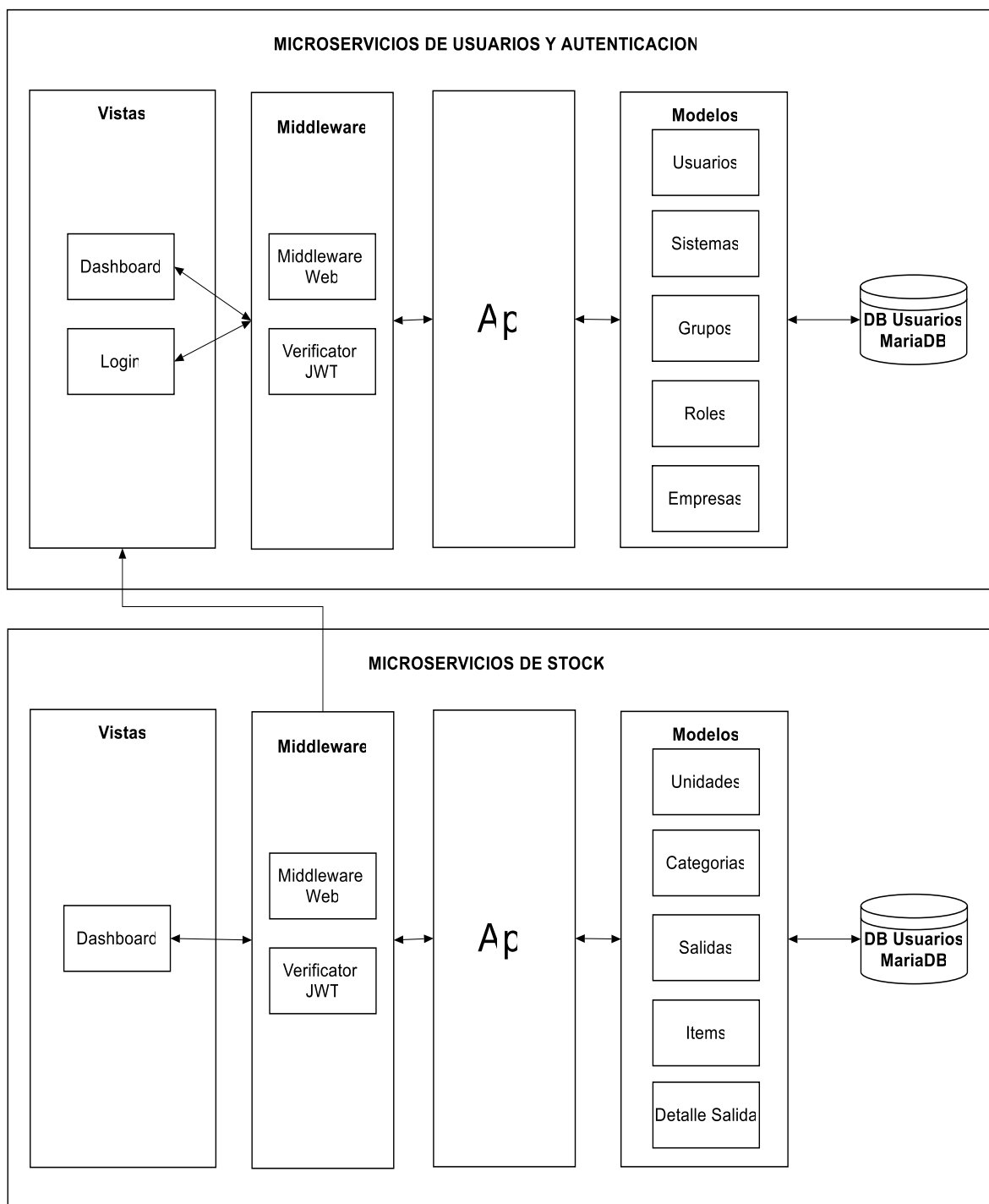
Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único del ítem.
nombre	Varchar	No	No	No	Nombre de la salida.
total	Decimal	No	No	No	Precio parcial.
fecha	Date	Si	No	No	Fecha Salida.
empresa_solicitante	Varchar	Si	No	No	Empresa que solicita ítems.

Fuente: [Elaboración propia]

3.3.2.3 Diseño de arquitectura de software.

Figura N° 3. 31

Arquitectura del sistema con el segundo incremento



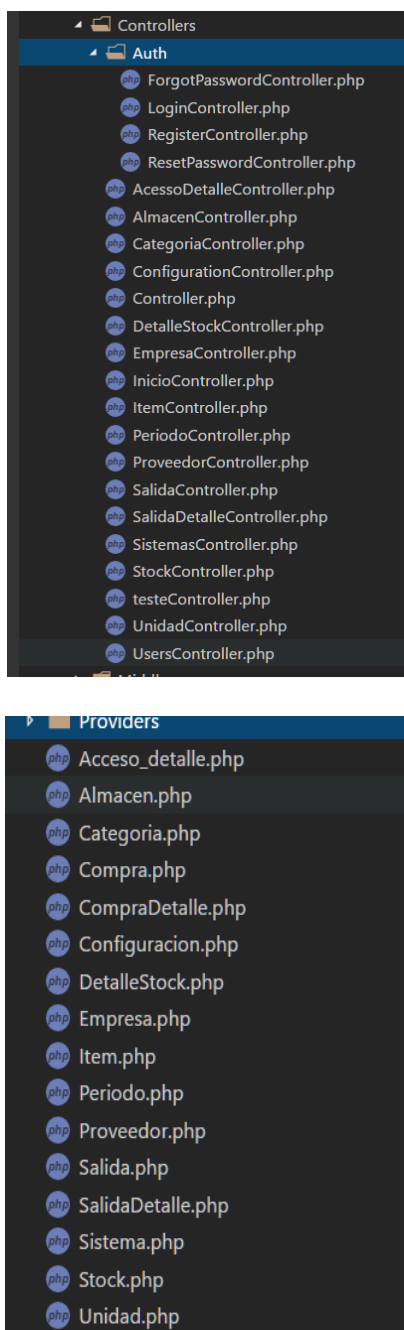
Fuente: [Elaboración propia]

3.3.3 Codificación segundo incremento

A continuación, se muestra la estructura de carpetas del subsistema stock.

Figura N° 3. 32

Estructura de carpetas de controladores y modelos stock



Fuente: [Elaboración propia]

Figura N° 3. 33

Codificación del procedimiento de salida de ítems

```
/*
CREATE PROCEDURE db_inventario.salidaItem(IN item_id INT, IN cantidad INT, IN status int,IN id_empresa INT)
    SQL SECURITY INVOKER
    READS SQL DATA
BEGIN
declare cantidad_items int;
declare verifyExistItemSalida int;
declare precioVenta decimal(10,2);
DECLARE nombreProduct varchar(100);
/*temporal items */
create temporary table if not exists temp_item as (select * from items i where i.id=item_id);
/*set parameters of items */
set cantidad_items = (SELECT cantidad from temp_item);
set precioVenta = (SELECT precio_venta from temp_item);
set nombreProduct = (SELECT nombre from temp_item);
/*temporal SalidaDetalle */
create temporary table if not exists temp_salida as (select * from salida_detalle sds where sds.item_id=item_id and sds.status=0 AND sds.empresa_id=id_empresa);
set verifyExistItemSalida =(select count(id) from temp_salida);

IF cantidad_items >=cantidad then
    update items i set i.cantidad=i.cantidad-cantidad where i.id=item_id;
    IF verifyExistItemSalida = 0 then
        insert into salida_detalle(nombre_producto,cantidad,precio_venta,item_id,subTotal,status,empresa_id)
        VALUES (nombreProduct,cantidad,precioVenta,item_id,precioVenta*cantidad,status,id_empresa);
    ELSE
        UPDATE salida_detalle sde
        set
            sde.cantidad=sde.cantidad+cantidad,
            sde.precio_venta=precioVenta,
            sde.subTotal=precioVenta*cantidad
        WHERE sde.item_id=item_id;
    END if;
END if;
drop table if exists temp_item;
drop table if exists temp_salida;
END
```

Fuente: [Elaboración propia]

Figura N° 3. 34

Codificación del procediendo cerrar salida

```
CREATE PROCEDURE db_inventario.cerrarSalida(IN ordenSalida varchar(100), IN nombreCliente varchar(100),IN id_empresa int)
SQL SECURITY INVOKER
READS SQL DATA
BEGIN
declare totalsalida decimal(10,2);
declare idSalida int;
set totalsalida= (SELECT sum(subTotal) FROM salida_detalles sds WHERE sds.status=0 AND sds.empresa_id=id_empresa);
insert into salidas(nombre,total,fecha,empresa_solicitante,empresa_id)values(ordenSalida,totalsalida,DATE(NOW()),nombreCliente,id_empresa);
set idSalida=(SELECT s.id FROM salidas s ORDER BY s.id DESC LIMIT 1);

update salida_detalles sds
set
sds.salida_id=idSalida,
sds.status=1
WHERE
/* sds.salida_id=null
AND */
sds.status=0
AND
sds.empresa_id=id_empresa;

END
```

Fuente: [Elaboración propia]

3.3.3.1 Implementación y funcionalidades

Una vez agregado el módulo de stock en el segundo incremento, se procede a observar las interfaces que se realizó para este producto.

En las siguientes figuras veremos el módulo de registro de ítems para salidas de productos.

Figura N° 3. 35

Implementación y funcionalidad

The screenshot displays the 'Stock' application interface. On the left is a sidebar with navigation links: Categorías, Unidades, Salidas, Productos, and Empresa. The main area is divided into three panels:

- Datos Salida:** Contains input fields for 'Nombre Salida' (with the value 'Orden de Salida 2'), 'Empresa Solicitad' (with the value 'J.B.B.L.'), and a 'Cerrar' button at the bottom.
- Detalle Salida:** A table showing item details for 'FOCOS FLOURECENTE 10 WATTS'. The table has columns: Nombre Item, Cant. (5), P.Unit (10.00), and P.Subtotal (50.00). A 'Borrar' button is next to the item row. A 'TOTAL' row shows a subtotal of 50. Navigation controls (Previous, 1, Next) and a 'Cerrar' button are at the bottom.
- Productos disponibles:** A list of available products. It shows 'FOCOS FLOURECENTE 10 WATTS' with 'Cant. Disp.' of 10 and 'P. Venta' of 10.00. An 'Agregar' button is next to the item. Navigation controls (Previous, 1, Next) are at the bottom.

Fuente: [Elaboración propia]

En la siguiente figura se observa una tabla con ítems que tiene el subsistema de stock, incluye un buscador en tiempo real para la búsqueda de datos en tiempo real.

Figura N° 3. 36

Vista de ítems disponibles

Nombre	Cantidad	P. Unit.	P. Venta	Unidad	Categoría
ALICATE	10	25.00	30.00	u	Electricidad
CABLE	100	5.00	7.00	m	Electricidad
CEMENTO SOBOCE	10	80.00	90.00	u	Contruccion
CLAVOS	100	10.00	15.00	k	Revestimiento
FOCOS FLOURECENTE 10 WATTS	10	8.00	10.00	u	Interiores

Fuente: [Elaboración propia]

En la siguiente figura se observa una tabla con las unidades que tiene el subsistema de stock, incluye un buscador en tiempo real para la búsqueda de datos en tiempo real.

Figura N° 3. 37

Implementación y funcionalidad

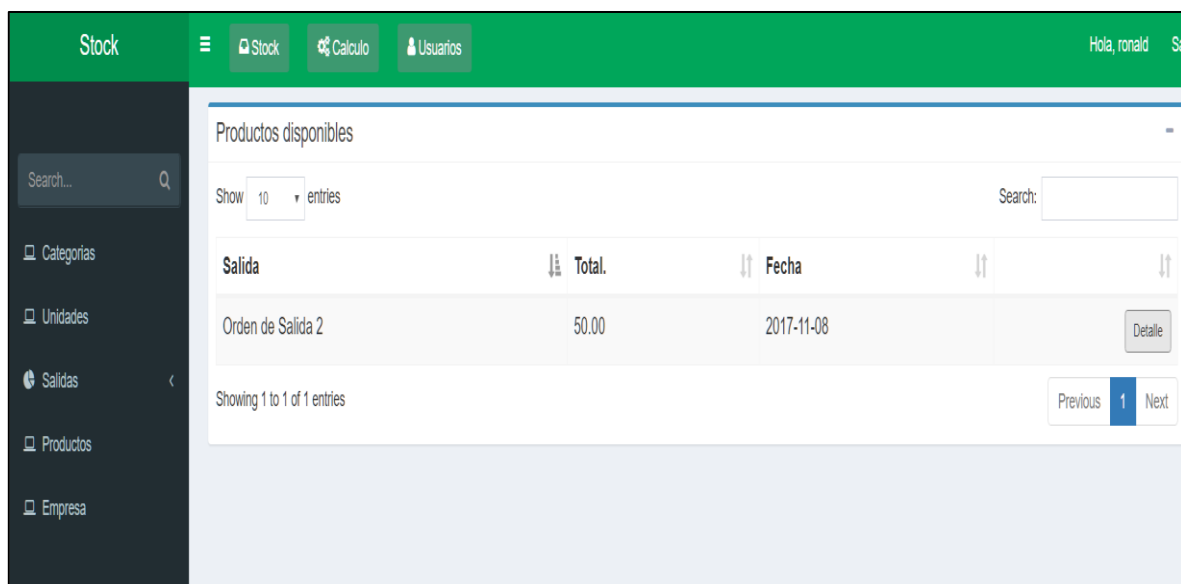
Nombre	Descripción
Cm	centimetros
g	gramos
k	kilogramos
lb	libras
m	metro
m2	metro cuadrado
m3	metro cubico
mm	milimetros
pz	piezas
u	unidades

Fuente: [Elaboración propia]

También se tiene la tabla de salidas incluyendo detalladamente por ítem en cada una de ellas.

Figura N° 3. 38

Implementación y funcionalidad



The screenshot shows a web application interface for 'Stock'. The top navigation bar is green and contains the title 'Stock', a menu icon, and buttons for 'Stock', 'Calculo', and 'Usuarios'. The user is logged in as 'Hola, ronald'. The left sidebar is dark grey and contains a search bar and a list of navigation items: 'Categorías', 'Unidades', 'Salidas' (selected), 'Productos', and 'Empresa'. The main content area is titled 'Productos disponibles' and shows a table with columns 'Salida', 'Total.', and 'Fecha'. The table contains one entry: 'Orden de Salida 2' with a total of 50.00 and a date of 2017-11-08. A 'Detalle' button is next to the entry. The table is paginated to show 1 to 1 of 1 entries.

Salida	Total.	Fecha
Orden de Salida 2	50.00	2017-11-08



The screenshot shows a modal window titled 'Detalle'. It displays the following information: 'Nombre Salida: Orden de Salida 2', 'Empresa Solicitud: J.B.B.L.', and 'Fecha Salida: 2017-11-08'. Below this is a table with columns 'Nombre Producto', 'Cantidad', 'P. Unit', and 'Sub Total'. The table contains two rows: one for 'FOCOS FLOURECENTE 10 WATTS' with a quantity of 5, unit price of 10.00, and sub-total of 50.00; and a 'TOTAL' row with a sub-total of 50.00. A 'Cerrar' button is at the bottom right.

Nombre Producto	Cantidad	P. Unit	Sub Total
FOCOS FLOURECENTE 10 WATTS	5	10.00	50.00
		TOTAL	50.00

Fuente: [Elaboración propia]

3.3.4 Casos de prueba segundo incremento

Para una entrega de calidad se procede a realizar pruebas de funcionalidad tomando en cuenta las acciones que se realiza en las vistas de mayor impacto en el proyecto.

Tabla N° 3. 25

Casos de prueba realizados para el segundo incremento

Caso de prueba	Descripción	Prueba Realizada	Resultado Esperado
Ingresar Producto.	Se registrará nuevos ítems.	Cuando agregue un nuevo ítem este deberá verificar que no haya campos vacíos.	Verifica y muestra un mensaje de validación indicando que no debe ir campos vacíos. [Correcto]
		Se intento agregar un ítem existente.	Verifica y muestra mensaje de validación indicando la existencia del ítem. [Correcto]
		Se registra el ítem.	Registra el ítem y muestra un mensaje indicando que el ítem se almaceno en la base de datos.
		Debe buscar el ítem necesario y si existe debe agregar a la tabla de salida de ítems.	En caso de que el ítem exista muestra una opción para agregar y la cantidad a ser agregada a la tabla de salida. [Correcto]

Registrar Salida.	Se registrará la salida y cantidad de ítems.	En caso de que exista ítems en la tabla salida y se ingresa de nuevo los datos deben actualizarse.	Al agregar a la tabla salida la tabla se actualiza y no permite ítems duplicados. [Correcto]
		A la salida de ítems se debe actualizar el campo de cantidad existente descontando cuanto salió.	El sistema actualiza la tabla de cantidad de ítems cuando son agregadas a la tabla salida. [Correcto]
		Debe mostrar el monto total de la tabla de salida.	Existe un campo donde se actualiza cada vez que se hace una salida de ítems. [Correcto]
Cerrar Salidas.	Cerrar los ítems de salidas.	Una vez terminada la selección de ítems para la tabla salida cerrar esta.	Para cerrar es necesario poner el nombre de la salida y la empresa que la está solicitando. [Correcto]
Mostrar Detalle de salidas de ítems.	Detallar todos los ítems que salieron de stock.	Se debe listar todas las salidas con sus respectivos detalles.	Las salidas que fueron cerradas se muestran en una tabla con una opción de 'Detalle' para listar los ítems que salieron de stock.

			[Correcto]
--	--	--	------------

Fuente: [Elaboración propia]

3.4 Tercer Incremento: Desarrollar el subsistema de cálculo para el sistema web.

En este incremento se hará el análisis, elaboración de diagramas, diseño de la base de datos, la codificación y los casos de prueba que sean necesarios.

3.4.1 Análisis de requerimientos del tercer incremento

Requerimientos funcionales:

- El administrador podrá crear, modificar, buscar, eliminar parámetros.
- El administrador podrá crear, modificar, buscar, eliminar formulas.
- El administrador podrá crear, modificar, buscar, eliminar módulos.
- El administrador podrá crear, modificar, buscar, eliminar proyectos.

Requerimientos no funcionales

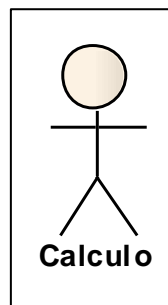
- El sistema deberá funcionar mediante un JWT generado.
- El JWT le permitirá acceso al subsistema y rechazará el ingreso no autorizado.

3.4.1.1 Identificación y descripción de actores

Se identifico el actor de cálculo el cual será representado en los diagramas como se muestra en la siguiente figura.

Figura N° 3. 39

Actor tercer incremento



Fuente: [Elaboración propia]

3.4.1.2 Identificación de casos de uso por actor

A continuación, se muestra los casos de uso por actor

Figura N° 3. 40

Diagrama de caso de uso del subsistema de cálculo



Fuente: [Elaboración propia]

3.4.1.3 Diagramas de caso de uso por stock

Caso de uso: Gestionar parámetros

Permite registrar nuevos parámetros que serán usados como para crear las fórmulas.

Tabla N° 3. 26

Caso de uso: Gestionar parámetros

Propósito:	Registro de nuevos parámetros para crear generar formulas.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.• Servicio de ítems del subsistema de stock.
Postcondiciones:	Almacenamiento en la base de datos de cálculo.
Circunstancias de uso:	Restringido solo para administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar formulas

Permite registrar nuevas fórmulas que serán usados en distintos módulos.

Tabla N° 3. 27

Caso de uso: Gestionar formulas

Propósito:	Registro de nuevas fórmulas para el cálculo de costo de módulos en base a parámetros.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.• Registro de unidades de ítems de stock.• Registro de categoría.

	<ul style="list-style-type: none"> • Validar datos de ingreso.
Postcondiciones:	Almacenamiento en la base de datos de cálculo.
Circunstancias de uso:	Restringido solo para administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar módulos

Permite registrar nuevos módulos que serán usados como para crear los proyectos.

Tabla N° 3. 28

Caso de uso: Gestionar módulos

Propósito:	Registro de nuevos módulos que serán usados en el proyecto.
Precondiciones:	<ul style="list-style-type: none"> • Conexión a la base de datos. • Tener privilegios para administrar el sistema. • Registro de fórmulas. • Validación de datos.
Postcondiciones:	Almacenamiento en la base de datos de cálculo.
Circunstancias de uso:	Restringido solo para administradores del sistema.

Fuente: [Elaboración propia]

Caso de uso: Gestionar proyectos

Permite registrar nuevos proyectos para su cotización

Tabla N° 3. 29

Caso de uso: Gestionar proyectos

Propósito:	La aplicación registrara los proyectos con el coste total detallado.
-------------------	--

Precondiciones:	<ul style="list-style-type: none"> • Conexión a la base de datos • Tener privilegios para administrar el sistema • Registro de módulos • Registro de categorías • Validación de datos.
Postcondiciones:	Almacenamiento en la base de datos de cálculo
Circunstancias de uso:	Restringido solo para administradores del sistema

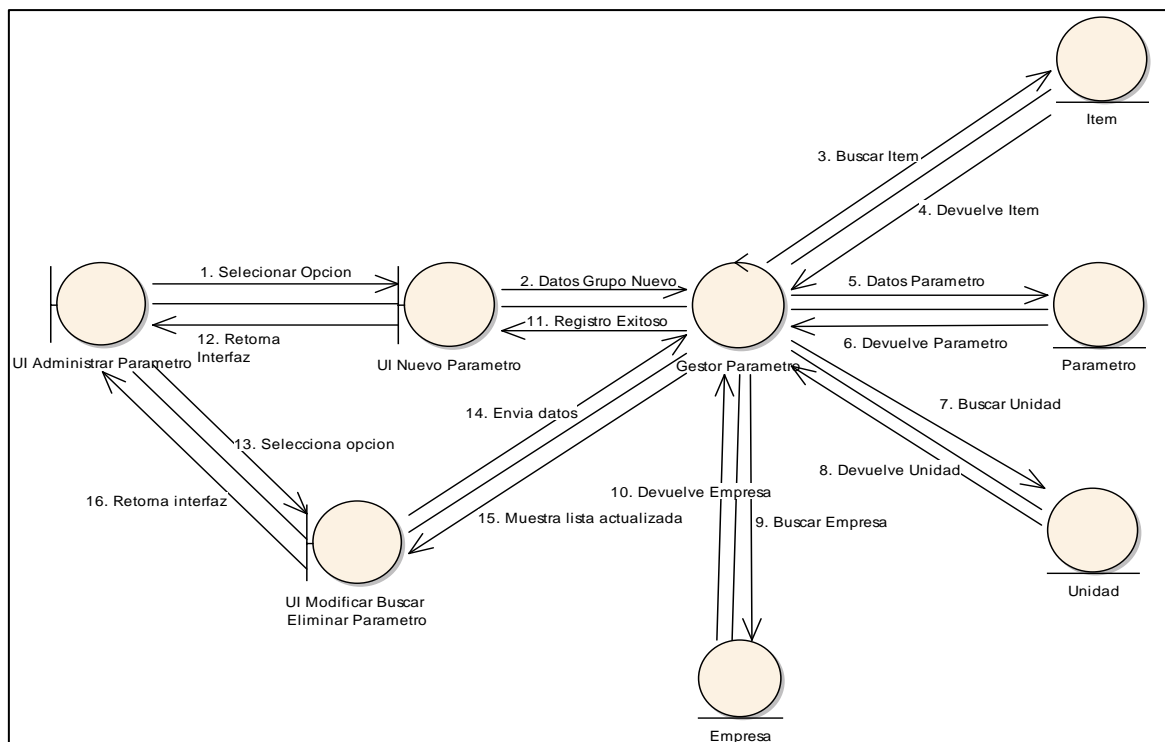
Fuente: [Elaboración propia]

3.4.1.4 Diagramas de colaboración del subsistema de cálculo

En las siguientes figuras se mostrarán los diagramas de colaboración realizados para los casos de uso del subsistema de cálculo.

Figura N° 3. 41

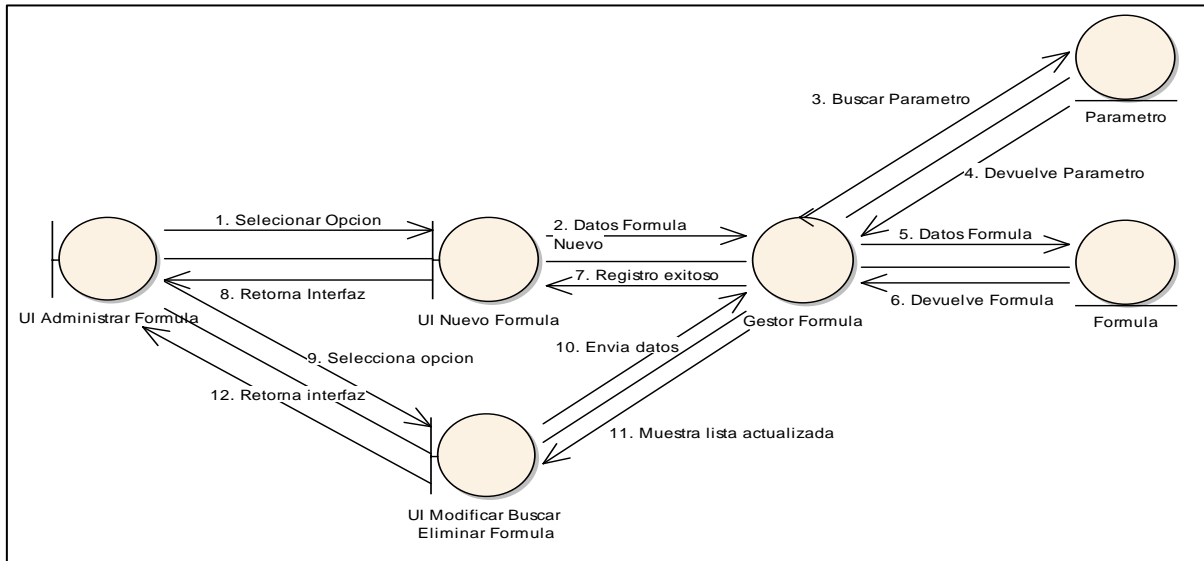
Diagramas de colaboración: Parámetros



Fuente: [Elaboración propia]

Figura N° 3. 42

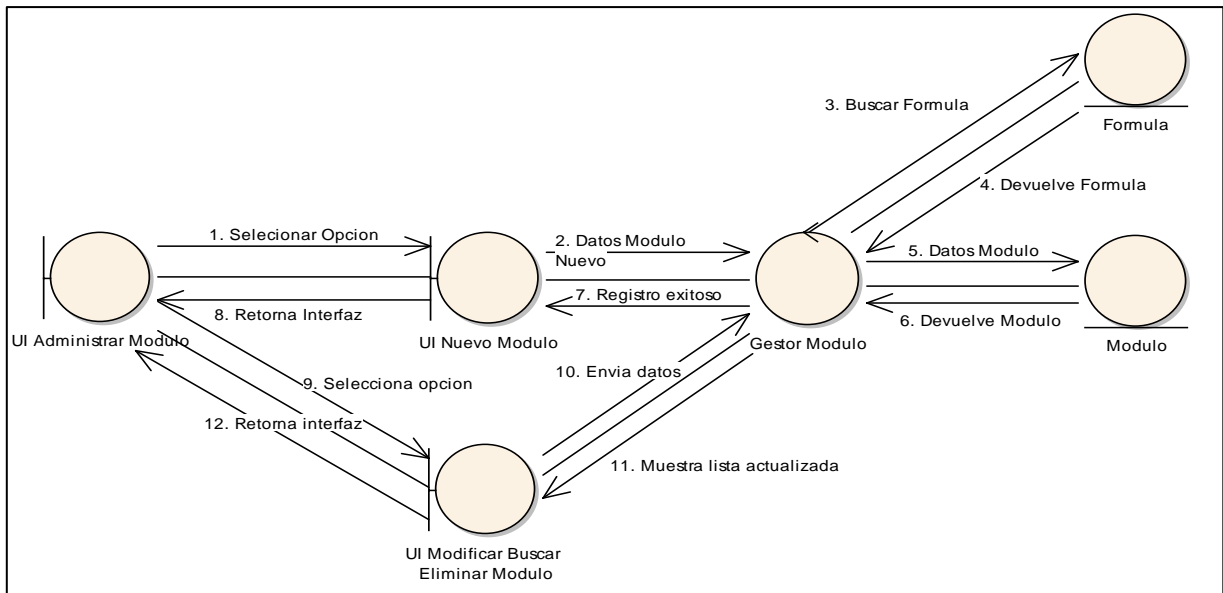
Diagramas de colaboración formulas



Fuente: [Elaboración propia]

Figura N° 3. 43

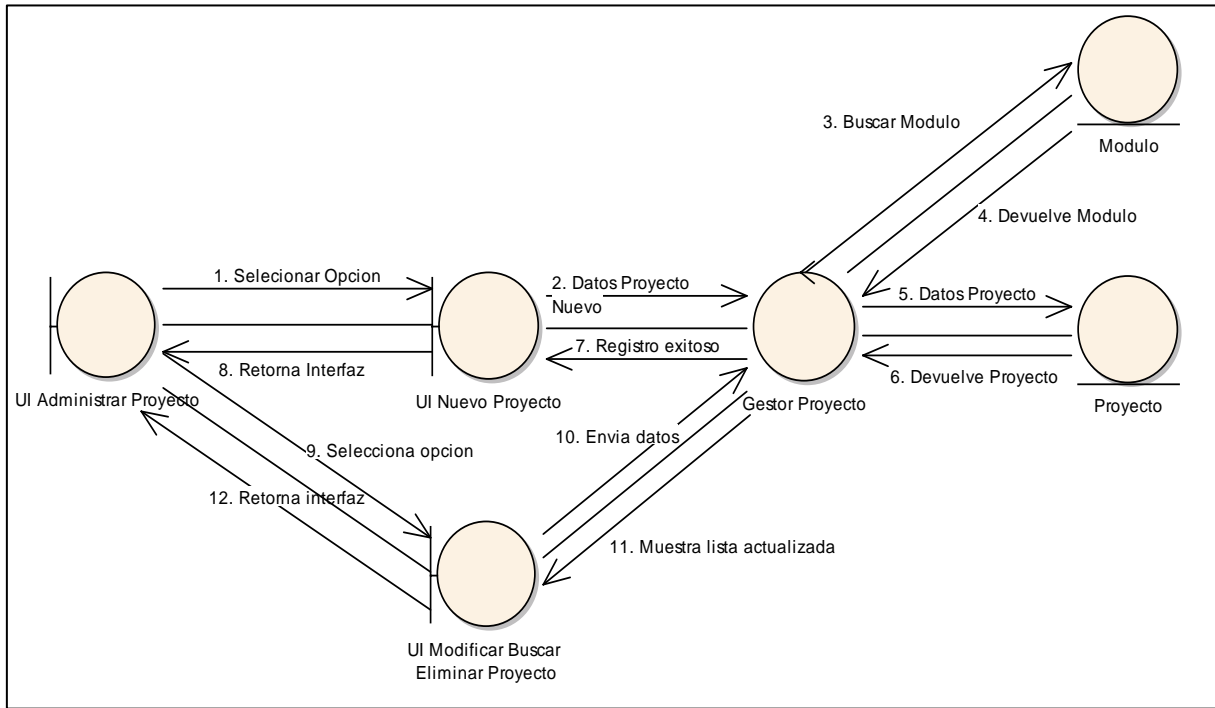
Diagramas de colaboración módulos



Fuente: [Elaboración propia]

Figura N° 3. 44

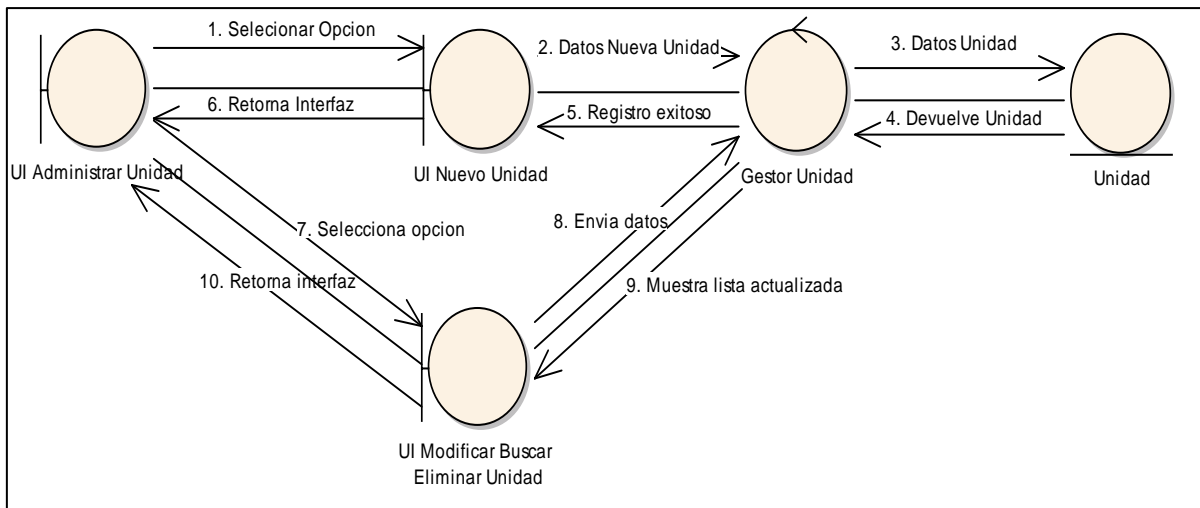
Diagramas de colaboración proyecto



Fuente: [Elaboración propia]

Figura N° 3. 45

Diagramas de colaboración unidades

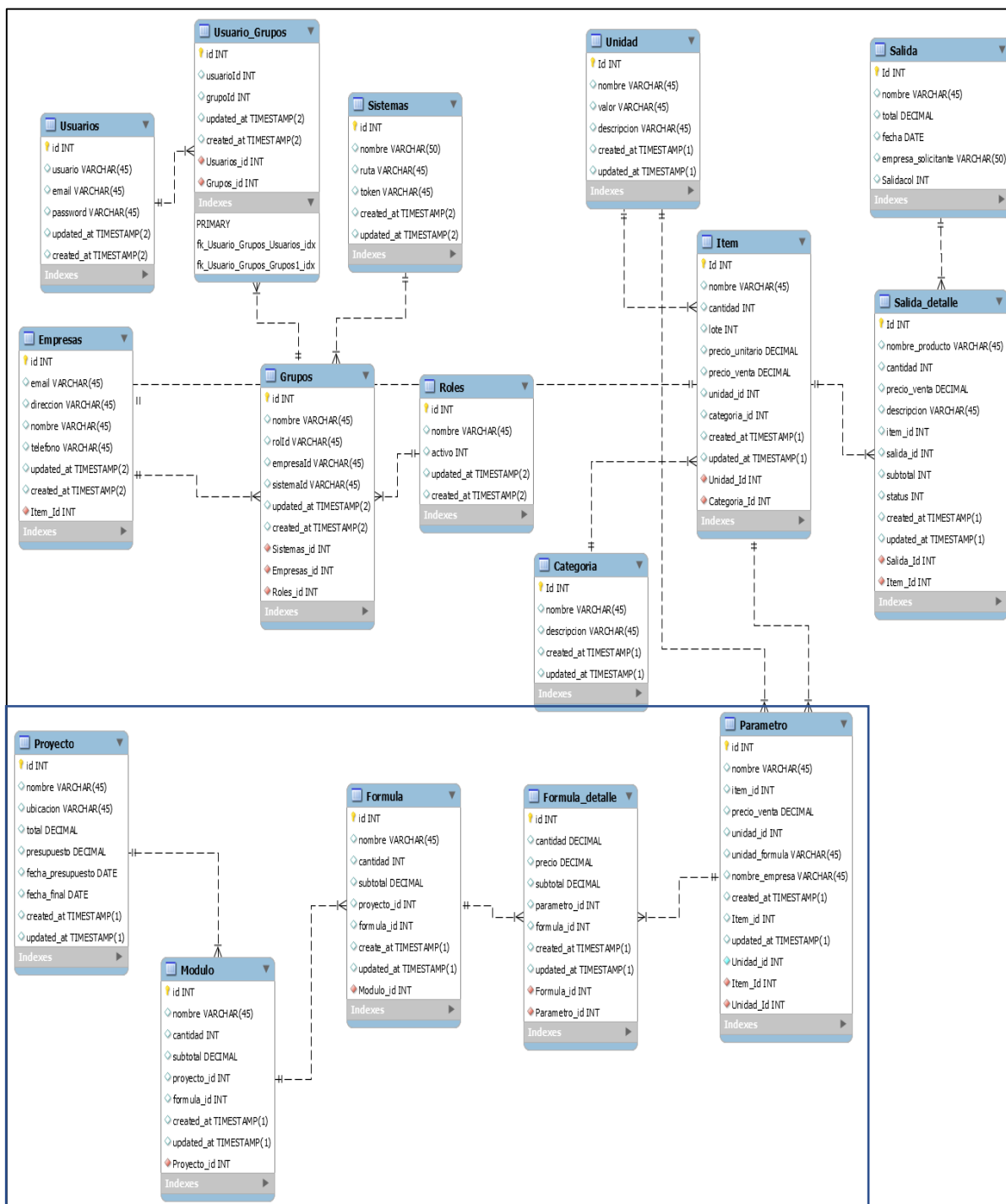


Fuente: [Elaboración propia]

3.4.2.2 Diseño de base de datos

Figura N° 3. 47

Base de datos para el tercer incremento



Fuente: [Elaboración propia]

3.4.2.3 Diccionario de datos

Tabla N° 3. 30

Diccionario de datos de la tabla Parámetro

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el id del parámetro.
nombre	Varchar	No	No	No	Registra el nombre del parámetro.
precio_venta	Decimal	No	No	No	Registra el precio que se adquirirá el ítem.
unidad_id	Int	No	No	Si	Registra el id de la unidad.
empresa_id	Int	No	No	Si	Registra el id de la empresa.
nombre_empresa	Varchar	No	No	No	Registra el nombre de la empresa.
unidad_formula	Varchar	No	No	No	Registra el tipo de unidad para la formula.
ítem_id	Int	No	No	Si	Registra el id de ítem.

Fuente: [Elaboración propia]

Tabla N° 3. 31

Diccionario de datos de la tabla Formula_detalle

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
Id	Int	No	Si	No	Registra el id único de la Formula.
nombre	Varchar	No	No	No	Registra el nombre de la empresa.
cantidad	Varchar	No	No	No	Registra la cantidad de la formula
precio	Varchar	No	No	No	Valor de la unidad.
subtotal	Decimal	No	No	No	Registra el subtotal calculado.
parámetro_id	Int	No	No	Si	Registra el id del parámetro.
formula_id	Int	No	No	Si	Registra el id de la formula.

Fuente: [Elaboración propia]

Tabla N° 3. 32

Diccionario de datos de la tabla Formula

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el id único de la formula.
nombre	Varchar	No	No	No	Registra el nombre de la formula.
Fecha	Date	No	No	No	Registra la fecha de creación de la formula.
subtotal	decimal	No	No	No	Registral el subtotal generado de la formula.

Fuente: [Elaboración propia]

Tabla N° 3. 33

Diccionario de datos de la tabla Módulo

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el id del módulo.

nombre	Varchar	No	No	No	Registra el nombre del módulo.
cantidad	Decimal	No	No	No	Registra la cantidad requerida.
subtotal	Decimal	No	No	No	Ingresa el costo total del módulo.
proyecto_id	Int	No	No	Si	Ingresa el id del proyecto.

Fuente: [Elaboración propia]

Tabla N° 3. 34

Diccionario de datos de la tabla Proyecto

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el id único del proyecto.
nombre	Varchar	No	No	No	Registra el nombre del proyecto.
ubicación	Varchar	No	No	No	Registra la ubicación del proyecto.

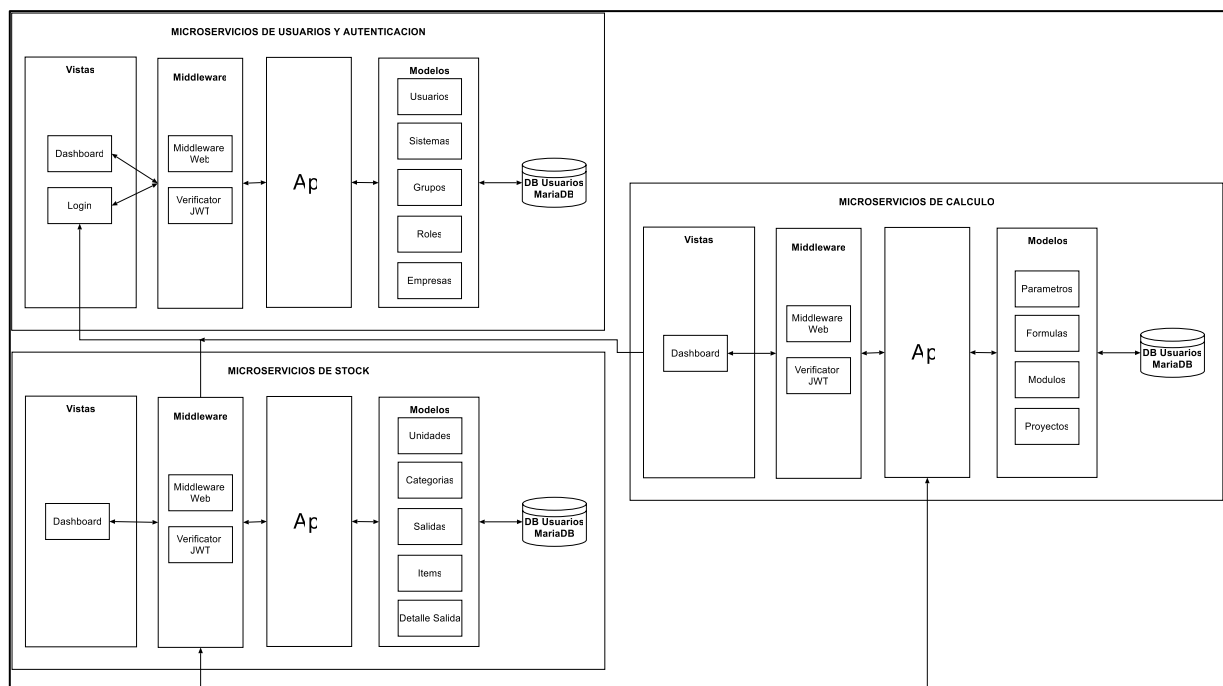
total	Decimal	No	No	No	Registra el costo total del proyecto.
presupuesto	Decimal	No	No	No	Presupuesto para la obra.
fecha_presupuesto	Date	No	No	No	Fecha generación de presupuesto del proyecto.
fecha_final	Date	No	No	No	Fecha final del proyecto.

Fuente: [Elaboración propia]

3.4.2.4 Diseño de arquitectura de software

Figura N° 3. 48

Diseño de la arquitectura tercer incremento



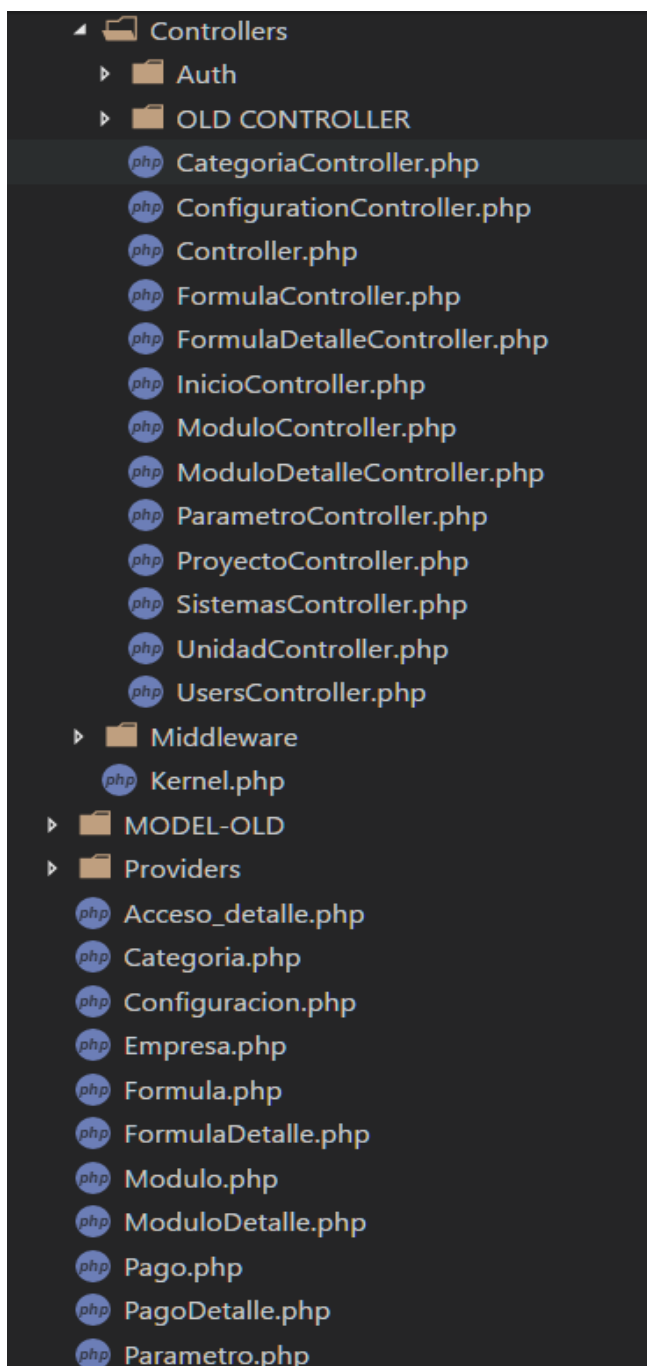
Fuente: [Elaboración propia]

3.4.3 Codificación del tercer incremento

A continuación, se muestra la organización de carpetas del subsistema calculo.

Figura N° 3. 49

Estructura de carpetas de controladores y modelos cálculo



Fuente: [Elaboración propia]

3.4.3.1 Implementación y funcionalidades

A continuación, se muestran algunas capturas de pantalla del funcionamiento del sistema.

Figura N° 3. 50

Creación de parámetros

Parametros			
Nombre Producto	P. Unit.		
ALAMBRE DE AMARRE	12.50		Borrar
ALAMBRE DE AMARRE	12.00		Borrar
ALQUITRAN	12.00		Borrar
ARENA COMUN	125.00		Borrar
CEMENTO PORTLAND	57.50		Borrar
CLAVOS	12.00		Borrar
CLAVOS	15.00		Borrar
ESPECIALISTA	10.00		Borrar

Productos disponibles			
Nombre	Cantidad	Precio	
ALAMBRE DE AMARRE	100	12.50	Agregar
ALAMBRE DE AMARRE	100	12.00	Agregar
ALQUITRAN	100	12.00	Agregar
ARENA COMUN	95	125.00	Agregar
ARENA FINA	100	135.00	Agregar
ARENA PARA ENLOSETADO (CAPA BASE)	100	110.00	Agregar
AYUDANTE	100	11.20	Agregar
CEMENTO PORTLAND	95	57.50	Agregar
CLAVOS	100	15.00	Agregar
CLAVOS	100	12.00	Agregar

Fuente: [Elaboración propia]

Figura N° 3. 51

Vista de formulas

nombre formula	Total.	Fecha	
Pared 3x3x3	62.50	11/11/2017	Detalle
Piscina	182.50	11/10/2017	Detalle
Pisos 1x1x1	0.00	11/16/2017	Detalle
Techo	85.10	11/08/2017	Detalle

Fuente: [Elaboración propia]

Figura N° 3. 52

Detalle de formulas

Detalle

Nombre Salida: Piscina
Fecha Salida: 11/10/2017

Nombre Producto	Unidad	Cant.	P. Unit.	SubTotal
CEMENTO PORTLAND	KG	50.000	1.15	57.50
			TOTAL	182.50

Cerrar

Fuente: [Elaboración propia]

Figura N° 3. 53

Creación de módulos

Calculo

Stock
Calculo
Usuarios
Pagos
Fases
Salir

Search...

Parametros
Formulas
Crear Formulas
Proyectos
Modulos

Datos Modulo

Nombre Modulo
Pared 5
Crear

Detalle Modulo

Show 10 entries
Search:

N. Calculo	cantidad	Precio	Sub Total
Pared 3x3x3	5.00	62.50	312.50
		TOTAL	312.5

Showing 1 to 1 of 1 entries

Previous 1 Next

Formula Disponibles

Show 10 entries
Search:

Nombre	Precio
Pared 3x3x3	62.50
Piscina	182.50
Pisos 1x1x1	0.00
Techo	85.10

Showing 1 to 4 of 4 entries

Previous 1 Next

Fuente: [Elaboración propia]

Figura N° 3. 54

Nuevo Proyecto

Datos Proyecto

Nombre Proyecto

Enlosetado Av. Sanchez Lima


Ubicacion

La paz - Sopocachi

Presupuesto


120000

Fecha Inicial



10/04/2017

Fecha Final



11/30/2017

Crear

Fuente: [Elaboración propia]

3.4.4 Casos de prueba del tercer incremento

Se hizo las pruebas de funcionalidad al sistema, detallando en la siguiente tabla.

Tabla N° 3. 35

Casos de prueba para el tercer incremento

Caso de prueba	Descripción	Prueba Realizada	Resultado Esperado
		Deberá listar los ítems disponibles del subsistema de Stock.	Lista el detalle de los ítems disponibles. [Correcto]

Registrar nuevos parámetros.	Se registrará nuevos parámetros.	Se intento agregar un ítem existente.	No permite la duplicación de ítems. [Correcto]
Crear fórmula.	Se registrará la salida y cantidad de ítems.	Listar parámetros disponibles.	En una tabla muestra los parámetros disponibles. [Correcto]
		Deberá poder reutilizar los parámetros en distintas fórmulas.	Al registrar los parámetros en cada formula, estos siguen disponibles para su uso. [Correcto]
		Deberá mostrar la suma de los parámetros utilizados existente.	Cada vez que se agrega un nuevo parámetro, se suma automáticamente el precio total. [Correcto]
Crear Proyecto.	Deberá poder crear nuevos proyectos.	Se registro un proyecto en el sistema.	El sistema valido la existencia de un proyecto. [Correcto]

	Deberá listar los proyectos detalladamente.	Se listo la lista de proyectos existentes.	Detalla el proyecto por módulo y por fórmula. [Correcto]
Crear módulos.	Deberá permitir la creación de nuevos módulos.	Se creó distintos módulos para el proyecto.	Se pudo registrar los módulos necesarios para el proyecto. [Correcto]

Fuente: [Elaboración propia]

3.5 Cuarto Incremento: Desarrollar el subsistema de fases para el sistema web.

En esta fase se verificara el progreso de la obra civil.

3.5.1 Análisis de requerimientos del cuarto incremento

Requerimientos funcionales

- El administrador podrá crear, modificar, buscar, eliminar fases.

Requerimientos no funcionales

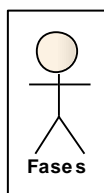
- Fácil e intuitivo

3.5.1.1 Identificación y descripción de actores

Se identifico el actor de fase el cual será representado en los diagramas como se muestra en la siguiente figura.

Figura N° 3. 55

Actor del cuarto incremento



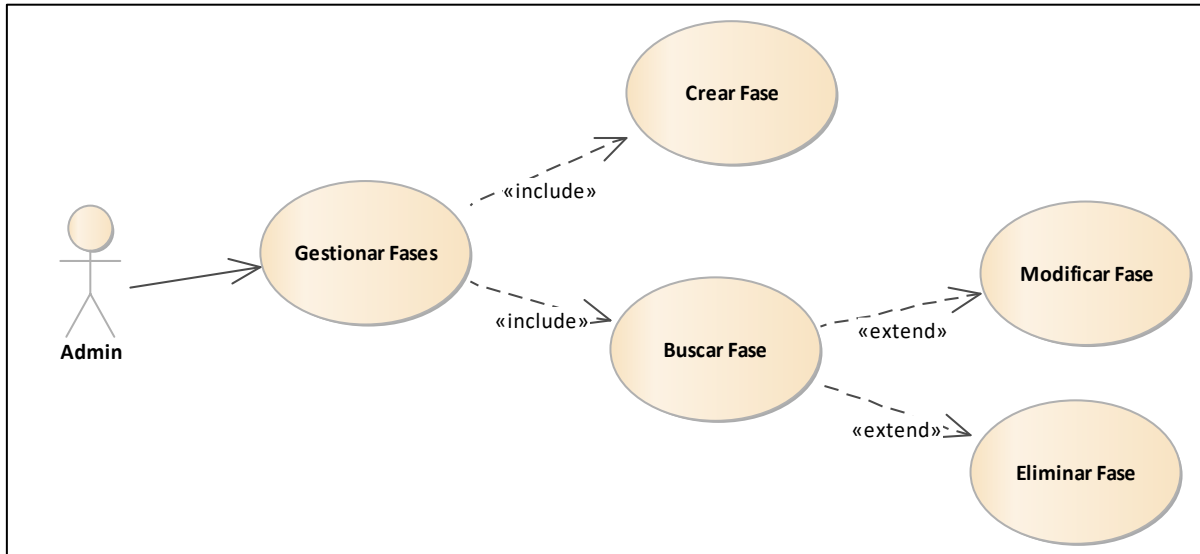
Fuente: [Elaboración propia]

3.5.1.2 Identificación de casos de uso por actor

A continuación, se muestra los casos de uso por actor

Figura N° 3. 56

Diagrama del caso de uso del subsistema de fase



Fuente: [Elaboración propia]

3.5.1.3 Diagramas de caso del cuarto incremento

Caso de uso: Gestionar fases

Permite administrar los tiempos de trabajo por modulo.

Tabla N° 3. 36

Caso de uso: Gestionar Fase

Propósito:	Registro de nuevas fases para tener los cronogramas de trabajo.
Precondiciones:	<ul style="list-style-type: none">• Conexión a la base de datos.• Tener privilegios para administrar el sistema.

	<ul style="list-style-type: none"> • Registro de proyecto.
Postcondiciones:	Almacenamiento en la base de datos de fases.
Circunstancias de uso:	Restringido solo para administradores del sistema.

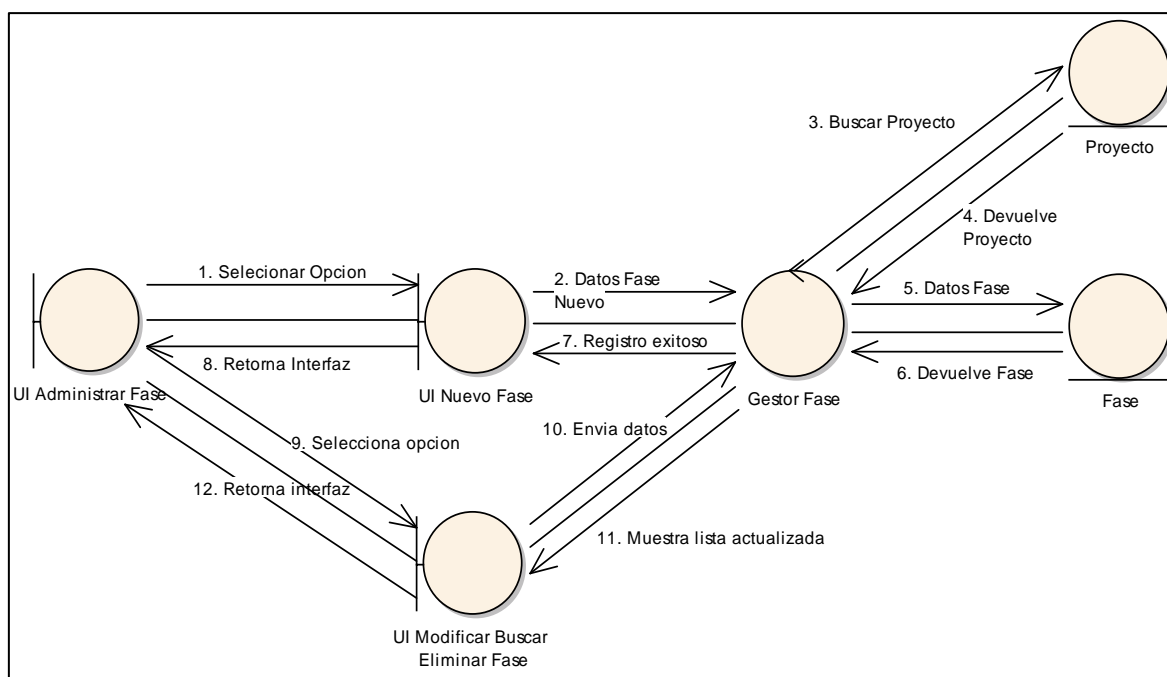
Fuente: [Elaboración propia]

3.5.1.4 Diagramas de colaboración de Fase

En el siguiente diagrama se muestra los diagramas de colaboración realizados para el subsistema de fases.

Figura N° 3. 57

Diagramas de colaboración: Fase



Fuente: [Elaboración propia]

3.5.2 Diseño de cuarto incremento

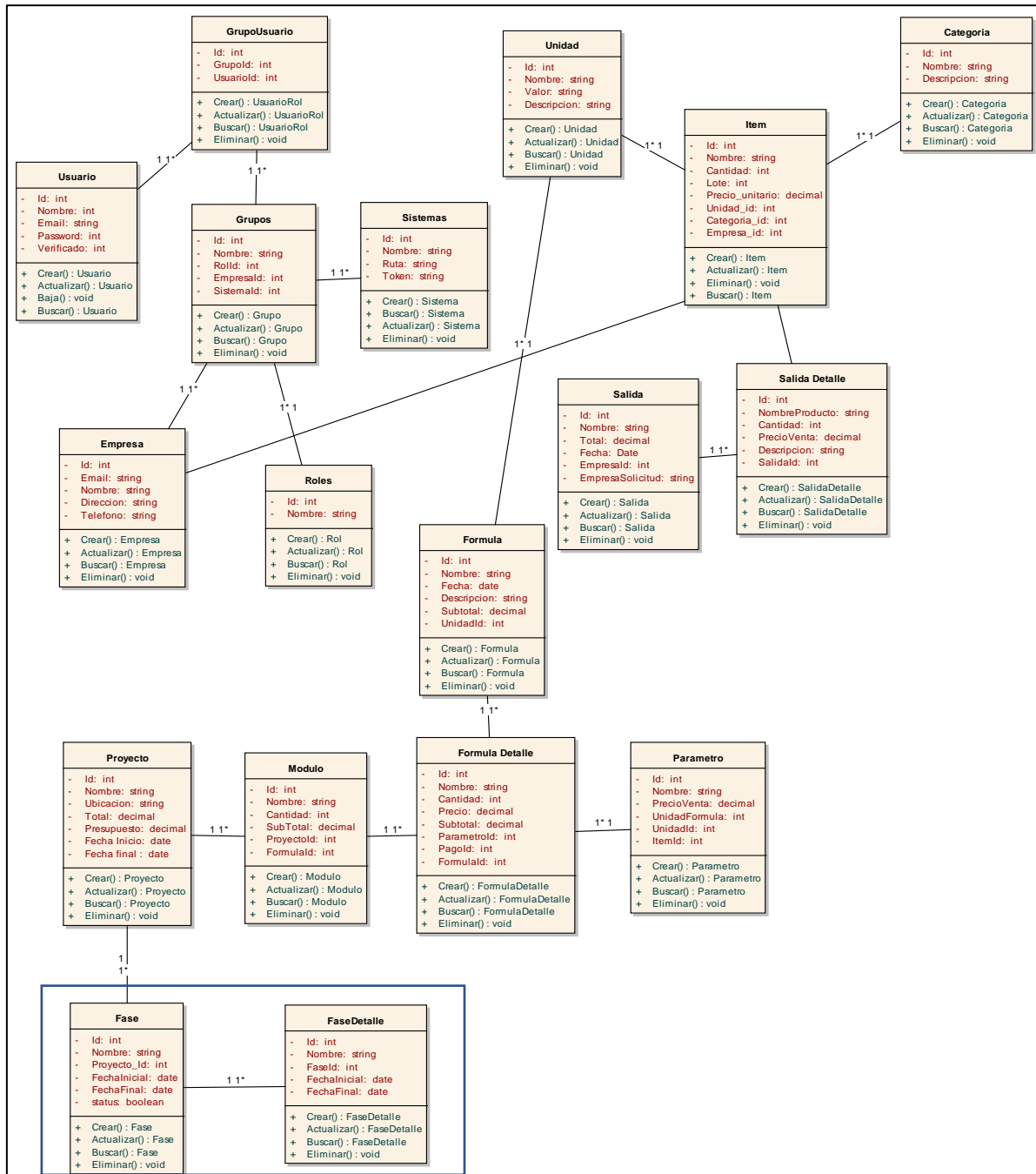
3.5.2.1 Diagrama de clases

A continuación, se muestra el diagrama de clases con el cuarto incremento del subsistema de fases.

Figura N° 3. 58

Diagrama de clases para el cuarto incremento

Para una mejor comprensión del incremento se incluyó el diagrama de clases y el de base de datos.

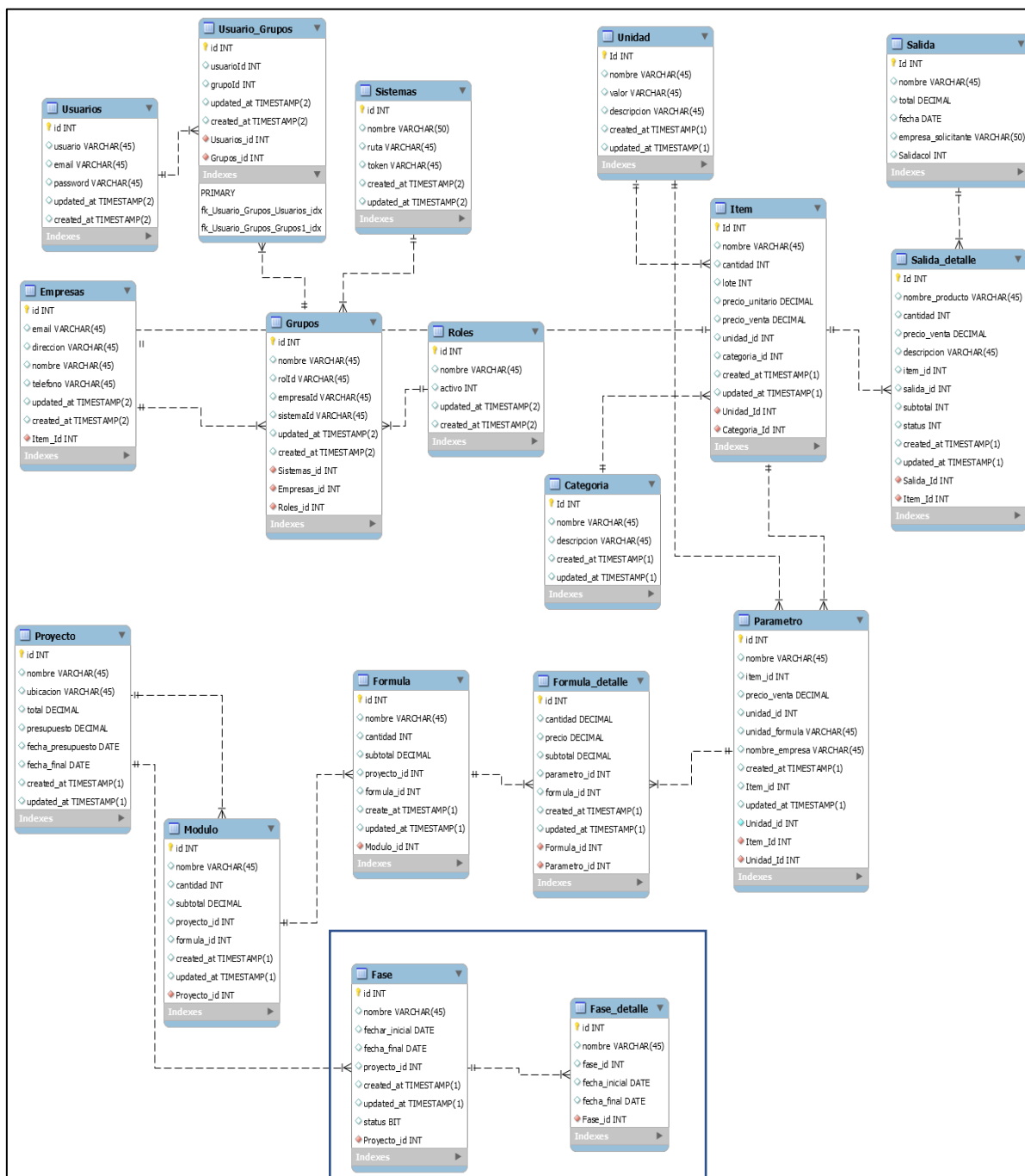


Fuente: [Elaboración propia]

3.5.2.2 Diseño de base de datos

Figura N° 3. 59

Base de datos para el cuarto incremento



Fuente: [Elaboración propia]

3.5.2.3 Diccionario de datos

En las siguientes tablas se describen todos los campos usados en la base de datos.

Tabla N° 3. 37

Diccionario de datos de la tabla fases

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código único de la fase.
nombre	Varchar	No	No	No	Registra el nombre de la fase.
fecha_inicial	Date	No	No	No	Fecha inicial de ejecución.
fecha_final	Date	No	No	No	Fecha conclusión.
proyecto_id	Int	No	No	Si	Código de proyecto.
status	Int	Si	No	No	Estado de la fase.

Fuente: [Elaboración propia]

Tabla N° 3. 38

Diccionario de datos de la tabla fases_detalle

Columna	Tipo de dato	Null	Llave primaria	Llave foránea	Descripción
id	Int	No	Si	No	Registra el código de fase_detalle.
nombre	Varchar	No	No	No	Registra el nombre de subfase.
fase_id	Int	No	No	Si	Código de la fase
fecha_inicial	Date	No	No	No	Fecha inicial de la subfase.
fecha_final	Date	No	No	No	Fecha final de la subfase.
modulo_id	Int	No	No	Si	Código del módulo.

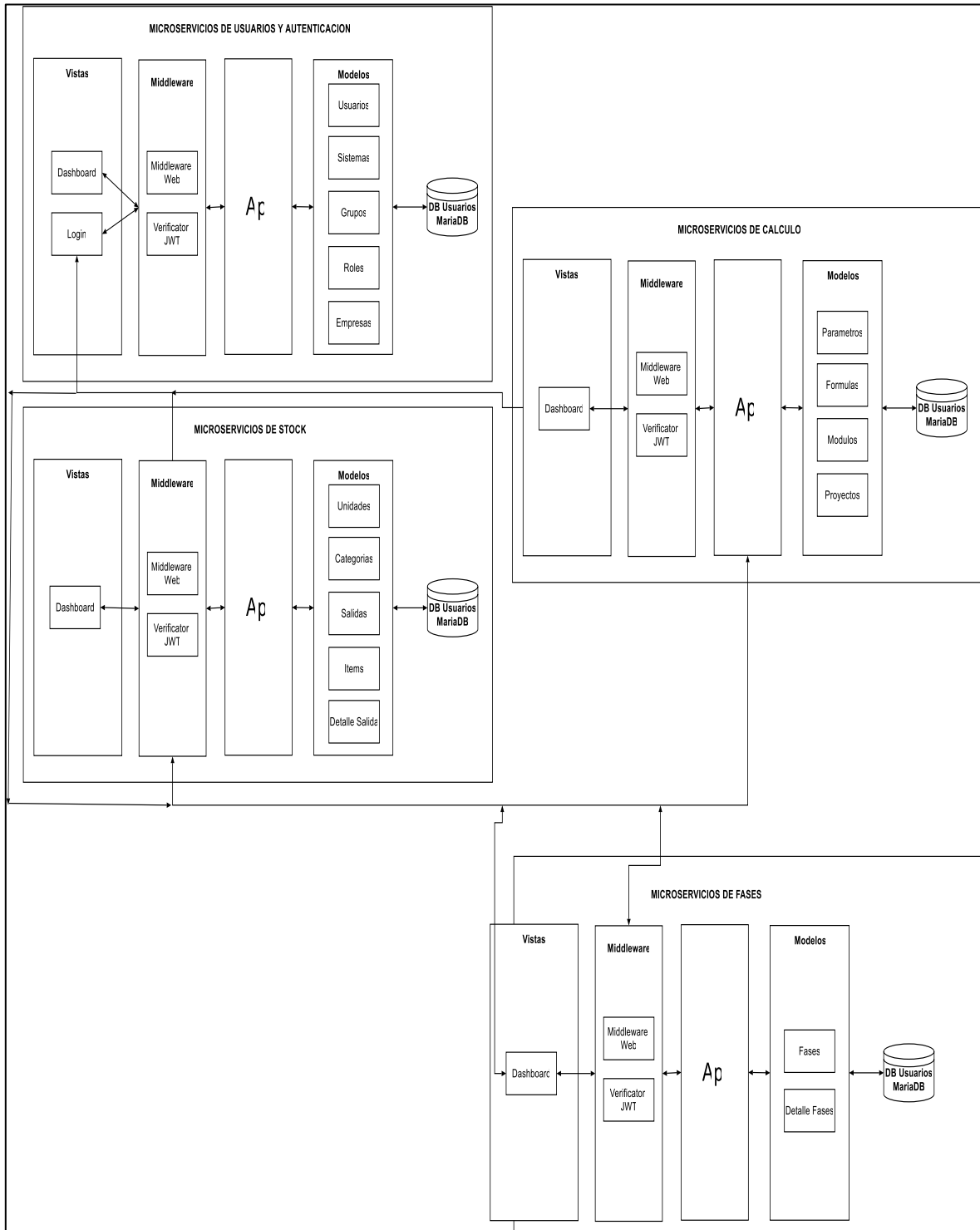
Fuente: [Elaboración propia]

3.5.2.4 Diseño de arquitectura de software

Se diseñó el incremento en la arquitectura para la cuarta iteración como es mostrada en la siguiente figura.

Figura N° 3. 60

Arquitectura para la cuarta iteración



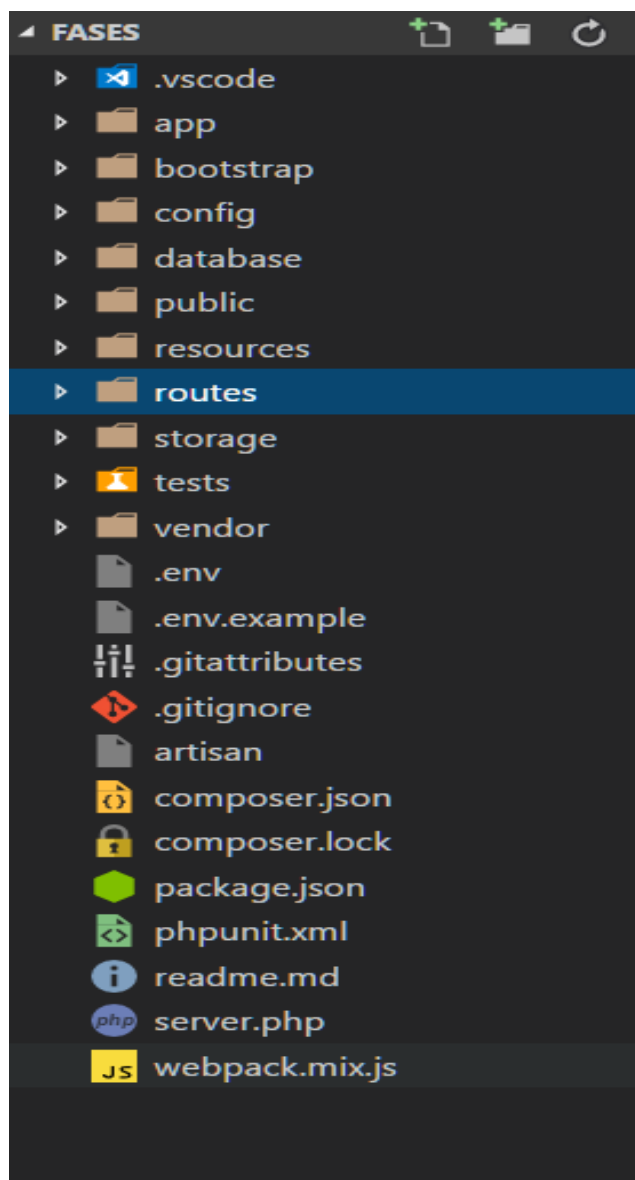
Fuente: [Elaboración propia]

3.5.3 Codificación del cuarto incremento

A continuación, se muestra la estructura de carpetas del subsistema de fases.

Figura N° 3. 61

Organización de carpetas y archivos



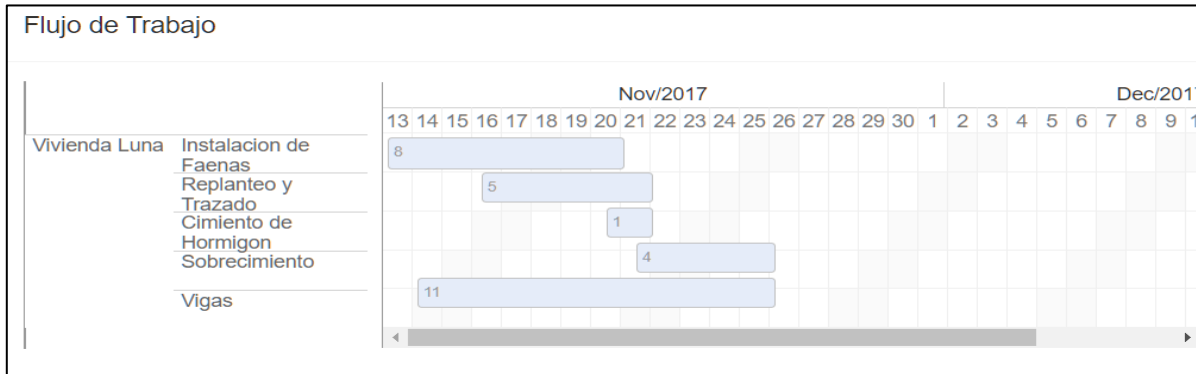
Fuente: [Elaboración propia]

3.5.3.1 Implementación y funcionalidades

En las siguientes figuras se muestra la funcionalidad del subsistema de fases.

Figura N° 3. 62

Diagrama Gantt generado por el sistema



Fuente: [Elaboración propia]

En la siguiente Figura se observa los proyectos creados por la empresa.

Figura N° 3. 63

Listado de proyectos de la empresa

Stock									
<div> <div>Stock</div> <div>Calculo</div> <div>Usuarios</div> <div>Pagos</div> <div>Fases</div> </div> <div>Hola, ronald</div> <div>Salir</div>									
Fases									
<div> <div>Search...</div> <div>Fases</div> </div>									
Nombre proyecto									
Vivienda Luna									
Enlosetado calle tupak katari									
Plaza triangular									
Cordon de acera									
Refaccion malaga									
PROYECTO PRUEBA									

Fuente: [Elaboración propia]

3.5.4 Casos de prueba del cuarto incremento

Para obtener un software de buena calidad es necesario hacer pruebas antes de llevarlos a producción.

Tabla N° 3. 39

Casos de prueba

Caso de prueba	Descripción	Prueba Realizada	Resultado Esperado
Listar Proyectos.	Deberá poder listar los proyectos del subsistema de cálculo.	Se ingreso a la pantalla de fases.	El sistema lista todos los proyectos del subsistema de cálculo. [Correcto]
	Listar detalladamente por modulo.	Se ingreso a la opción detalle de la vista pantalla fases.	Lista el proyecto en un diagrama GANTT por fase. [Correcto]
Modificar Fases.	Deberá poder modificar las fases en el diagrama GANTT.	Deberá permitir modificar las fechas de trabajo por fase.	El diagrama permite la modificación de fases por fechas. [Correcto]

Fuente: [Elaboración propia]

CONCLUSIONES

A continuación, se detallan las conclusiones que se obtuvieron en base al proyecto terminado.

Para iniciar el desarrollo del proyecto fue necesario realizar el modelo de negocio actual y alternativo, con una serie de entrevistas al dueño de la empresa. Todo esto ayudo a identificar las actividades del proceso de gestión de stock y cálculo de las obras que realiza la empresa.

Para el desarrollo del sistema se utilizó el framework Laravel y se publicó servicios. Cada subsistema tiene una base de datos propia.

- Todos los intercambios de los servicios fueron hechos bajo JSON ya que este aliviana el tamaño de datos haciéndolo más veloz.
- Para la integración de los subsistemas y el intercambio de datos sea seguro se usó el estándar JWT, permitiendo así la confidencialidad, integridad y la disponibilidad de los datos requeridos.
- Se propuso la arquitectura de microservicios porque permite una mayor escalabilidad del proyecto, es flexible en acoplamiento de servicios y se logró demostrar la funcionalidad de este.
- En el desarrollo del API del subsistema de autenticación y usuario se levantó un servidor propio el cual provee las credenciales y acceso a los demás subsistemas.
- En el desarrollo del módulo de stock se publicó las Apis para que se puedan consumir desde subsistemas internos o externos.
- El subsistema de cálculo fue el más moroso en realizarlo. Este tiene la característica de que todos los datos y formulas deben ser exactos.
- Para el desarrollo del subsistema de fases se incluyó el diagrama Gantt, en la cual se listan los proyectos realizados por la empresa, esta tiene un sencillo uso y tiene la característica de ser drag and drop.

RECOMENDACIONES

Se recomienda lo siguiente:

Para el funcionamiento del sistema se recomienda alquilar distintos hosts con php versión 7, en la cual estarán alojados los subsistemas.

Agregar autenticación por OAuth2 para que el usuario pueda registrarse desde una cuenta externa al sistema.

Para obtener un cálculo más preciso del presupuesto se recomienda agregar el subsistema de R.R.H.H.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Franco Y, (2011, junio 27). La entrevista. Disponible en:
<http://tesisdeinvestig.blogspot.com/2011/06/entrevistas.html>
- [2] Cerda Hugo, “Los elementos de la Investigación.” Bogotá, El Búho, 1991.
- [3] Procedia Manufacturing (2017) Disponible en:
<https://www.sciencedirect.com/science/article/pii/S2351978917300392/pdf?md5=95f9eb27c41680fdf17c332e2e663124&pid=1-s2.0-S2351978917300392-main.pdf>
- [4] Gutiérrez Maribel Sequeira, Guía para la Elaboración de Diagramas de Flujo, 2009.
- [5] Cuvadelcivil, Presupuestos de obra. Disponible en:
<http://www.cuvadelcivil.com/2010/06/presupuesto-de-obra.html>
- [6] Pressman, Roger, Ingeniería del software, México, D. F.: The McGraw-Hill Companies, Inc., 2010
- [7] Patponto, (2010, Septiembre 28). Ingeniería de Software.
Disponible en: <http://histinf.blogs.upv.es/2010/12/28/ingenieria-del-software/>
- [8] Centers for Medicare & Medicaid Services (2008, Marzo 27). Seleccionando un enfoque de desarrollo. Disponible en: <https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf>
- [9] Bermudez Cristian, Garrido Erika, Lara Natalia. Disponible en:
<https://procesossoftware.wikispaces.com/Modelo+Incremental>
- [10] Ecured. Modelo de Prototipos. Disponible en:
https://www.ecured.cu/Modelo_de_prototipos
- [11] Schmuller Joseph, Aprendiendo UML en 24 Horas. México: S.A. Alhambra Mexicana, 2000.
- [12] Van Der Henst Christian. (2001, Mayo 23). Introducción a Php. Disponible en:
<http://www.maestrosdelweb.com/phpintro/>

- [13] C. S. Roldán, (2013, Marzo 02). ¿Qué es Python? Disponible en: <https://www.codejobs.biz/es/blog/2013/03/02/que-es-python>
- [14] José Antonio González Seco (2001, Octubre 11). Qué es C#. Disponible en: <https://desarrolloweb.com/articulos/561.php>
- [15] Patricio, (2013, Marzo 21). Desarrollando Webs Dinámicas. Disponible en: <http://desarrollandowebsdinamicas.blogspot.com/2013/03/que-es-laravel.html>
- [16] Juan Miguel Vergara Pineda (2016, noviembre 24). Symfony: ¿Qué es y cuáles son sus principales características? Disponible en: <https://www.coriaweb.hosting/symfony-principales-caracteristicas>
- [17] Mkdocs, Qué es Yii? Disponible en: <https://yii2-framework.readthedocs.io/en/stable/guide-es/intro-yii>
- [18] S. Neuman, Building Microservices, Sebastopol: O'Reilly Media, Inc., 2015.
- [19] W. España, Guía Breve de Servicios Web. Disponible en: <https://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb>
- [20] BBVA (2016, Marzo 23). API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. Disponible en: <https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos>
- [21] Dept. Ciencia de la Computación e IA, (2017 Junio 26). Introducción a los Servicios Web. Invocación de servicios web SOAP. Disponible en: <http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html>
- [22] MariaDB. Disponible en: <https://mariadb.com/files/MariaDB.pdf>
- [23] Petkovic Dušan, "Microsoft SQL Server 2008", McGraw-Hill Interamericana, 2010.
- [24] JWT. Introduction to JSON Web Tokens. Disponible en: <https://jwt.io/introduction>
- [25] Programación Web (2018, Abril 14). Sesiones. Disponible en: <https://programacionwebisc.wordpress.com/3-9-sesiones/>

- [26] Mozilla, Cookies: información que los sitios web guardan en tu equipo. Disponible en: <https://support.mozilla.org/es/kb/cookies-informacion-que-los-sitios-web-guardan-en->
- [27] C. Blé, Diseño Ágil con TDD.
- [28] Daniel Diaz Suarez, (2013, Marzo 27). TDD, BDD & Test de Aceptación. Disponible en: <https://www.adictosaltrabajo.com/tutoriales/tdd-bdd-test-de-acceptacion>

ANEXOS

Anexo 1

Listado de Material

Listado precios de materiales

CODG	DESCRIPCION	UNIDAD	PRECIO MATERIAL (\$us)	SANTA CRUZ	LA PAZ y EL ALTO	COCHABAMBA	SUCRE	TARIJA
CD	DESCRIPCION	IT	UND					
M0814	ABRAZADERA 3/4"	pza	0,57	4,00	4,00	4,00	4,00	4,00
M0133	ABRAZADERA METALICA P/TUBERIA DE 4"	pza	2,44	17,00	17,00	17,00	17,00	17,00
M0476	ABRAZADERA PARA BAJANTE	pza	0,36	2,50	2,50	2,50	2,50	2,50
M0677	ACCESORIOS GALVANIZADOS D=1 1/2"	pza	1,15	8,00	8,00	8,00	8,00	8,00
M0317	ACCESORIOS GALVANIZADOS D=1"	pza	0,86	6,00	6,00	6,00	6,00	6,00
M0266	ACCESORIOS GALVANIZADOS D=1/2	pza	0,65	4,50	4,50	4,50	4,50	4,50
M0265	ACCESORIOS GALVANIZADOS D=3/4	pza	0,79	5,50	5,50	5,50	5,50	5,50
M0580	ACCESORIOS GALVANIZADOS PARA AGUA POTABLE	glb	35,92	250,00	250,00	250,00	250,00	250,00
M0219	ACCESORIOS PARA IABALINA	pza	7,90	55,00	55,00	55,00	55,00	55,00
M0669	ACCESORIOS PARA PARANTES RE VOLIBOL	par	11,49	80,00	80,00	80,00	80,00	80,00
M0602	ACCESORIOS PARA TANQUE DE AGUA	pza	17,24	120,00	120,00	120,00	120,00	120,00
M0743	ACCESORIOS PARA VIDRIO TEMPLADO 10mm	glb	29,45	205,00	205,00	205,00	205,00	205,00
M0579	ACCESORIOS PVC PARA DESAGUE	glb	35,92	250,00	250,00	250,00	250,00	250,00
M0142	ACEITE DE LINAZA (lt)	lt	3,16	22,00	22,00	22,00	22,00	22,00
M0762	ACEITE SUCIO	lt	0,36	2,50	2,50	2,50	2,50	2,50
M0749	ACIDO MURIATICO	lt	10,06	70,00	70,00	70,00	70,00	70,00
M0964	ADITIVO ACELERADOR DE FRAGUADO EXENTO DE CLORURO	kg	5,24	36,50	36,50	36,50	36,50	36,50
M0786	ADITIVO IMPERMEABILIZANTE DE FRAGUADO NORMAL	kg	2,08	14,50	14,50	14,50	14,50	14,50
M0940	ADITIVO PARA REDUCCION DE RESISTIBILIDAD DEL SUELO	kg	39,51	275,00	275,00	275,00	275,00	275,00
M0916	ADITIVO SUPER-PLASTIFICANTE (FLUIDIFICANTE)	kg	6,25	43,50	43,50	43,50	43,50	43,50
M0161	ADOBE DE TIERRA (10x20x40) cm	pza	0,07	0,50	0,50	0,50	0,50	0,50
M0345	ADOQUIN COMANCHE	pza	0,53	3,70	3,70	3,70	3,70	3,70
M0085	AGUA DE RED	m³	0,09	0,60	0,60	0,60	0,60	0,60
M0558	AGUA DE SISTERNA	m³	3,59	25,00	25,00	25,00	25,00	25,00
M0077	AGUARAS (lt)	lt	1,51	10,50	10,50	10,50	10,50	10,50
M0182	ALAMBRE AISLADO DE COBRE N° 10 AWG TW	m	0,90	6,25	6,25	6,25	6,25	6,25
M0116	ALAMBRE AISLADO DE COBRE N° 12 AWG TW	m	0,58	4,05	3,52	3,52	3,52	3,52
M0002	ALAMBRE AISLADO DE COBRE N° 14 AWG TW	m	0,42	2,93	2,25	2,25	2,25	2,25
M0383	ALAMBRE AISLADO DE COBRE N° 16 AWG TW	m	0,22	1,50	1,50	1,50	1,50	1,50
M0384	ALAMBRE AISLADO DE COBRE N° 18 AWG TW	m	0,14	0,98	0,98	0,98	0,98	0,98
M0793	ALAMBRE AISLADO DE COBRE N° 6 AWG TW	m	1,32	9,20	9,20	9,20	9,20	9,20
M0147	ALAMBRE AISLADO DE COBRE N° 8 AWG TW	m	1,25	8,70	8,70	8,70	8,70	8,70
M0001	ALAMBRE DE AMARRE N° 12 ó 16	kg	2,30	16,00	16,00	16,00	16,00	16,00
M0000	ALAMBRE DE COBRE RESISTIVO N° 3 AWG	m	14,15	90,50				

ANEXO 2

Presupuesto preparado en Microsoft Excel

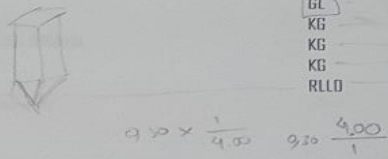
Detalle del presupuesto dividido en módulos.

Item: 0001 - INST. DE FAENAS Y COLOC. DE LETRERO OBRAS MENORES (VIAS)						1,00 GLS
Proyecto: CONST. ENLOSETADO CALLE TUPAC AMARU						
Nº	P.	Insumo/Parámetro	Und.	Cant.	Unit. (Bs)	Parcial (Bs)
	A	MATERIALES				1,740.00
1	-	DEPOSITO DE MATERIALES (ALQUILER)	PZA	2.00	350.00	700.00
2	-	OFICINA EN OBRA (ALQUILER)	PZA	1.00	250.00	250.00
3	-	LETRERO DE PANAFLEX CON ESTRUCTURA METALICA (FORMATO G.A.M.E.A.)	PZA	1.00	400.00	400.00
4	-	LETREROS DE PREVENCIÓN	PZA	2.00	70.00	140.00
5	-	HABITACION PARA SERENO (ALQUILER)	PZA	1.00	250.00	250.00
> D TOTAL MATERIALES					(A) =	1,740.00
	B	MANO DE OBRA				163.20
1	-	ALBAÑIL	hr	6.00	16.00	96.00
2	-	AYUDANTE	hr	6.00	11.20	67.20
> E SUBTOTAL MANO DE OBRA					(B) =	163.20
	F	Cargas Sociales		= 55.00% de	(E) =	89.76
	G	Impuesto al Valor Agregado		= 14.94% de	(E+F) =	37.79
> G TOTAL MANO DE OBRA					(E+F+G) =	290.75
	C	EQUIPO, MAQUINARIA Y HERRAMIENTAS				0.00
	H	Herramientas menores		6.00% de	(G) =	17.45
> I TOTAL HERRAMIENTAS Y EQUIPO					(C+H) =	17.45
> J SUB TOTAL					(D+G+I) =	2,048.20
	L	Gastos gales. y administrativ		8.00% de	(J) =	163.86

Anexo 3

Detalle presupuestado de la obra

>	D	TOTAL MATERIALES			(A) =	0.00
	B	MANO DE OBRA				19.63
1	-	OPERADOR DE COMPACTADORA	hr	0.35	17.50	6.13
2	-	PEON	hr	1.50	9.00	13.50
>	E	SUBTOTAL MANO DE OBRA			(B) =	19.63
	F	Cargas Sociales		55.00% de	(E) =	10.79
	D	Impuesto al Valor Agregado		14.94% de	(E+F) =	4.54
>	G	TOTAL MANO DE OBRA			(E+F+D) =	34.96
	C	EQUIPO, MAQUINARIA Y HERRAMIENTAS				14.00
1	-	COMPACTADOR TIPO SALTARIN	hr	0.35	40.00	14.00
	H	Herramientas menores		6.00% de	(G) =	2.10
>	I	TOTAL HERRAMIENTAS Y EQUIPO			(C+H) =	16.10
>	J	SUB TOTAL			(D+G+I) =	51.06
	L	Gastos grales. y administrativ		8.00% de	(J) =	4.08
	M	Utilidad		8.00% de	(J+L) =	4.41

	M	Utilidad		8.00% de	(J+L) =	176.96
>	N	PARCIAL			(J+L+M)	2.388,02
	P	Impuesto a las Transacciones		3.09% de	(N) =	73.82
>	Q	TOTAL PRECIO UNITARIO			(N+P) =	2.462.84
>		PRECIO ADOPTADO:				2.462.84
		Son: Dos Mil Cuatrocientos Sesenta y Dos con 84/100 Bolivianos				
		Item: 0002 - REPLANTEO Y TRAZADO DE VIAS				123.23 ML
		Proyecto: CONST. ENLOSETADO CALLE TUPAC AMARU				
Nº	P.	Insumo/Parámetro	Und.	Cant.	Unit. (Bs)	Parcial (Bs)
A		MATERIALES				2.37
1	-	ESTACAS (2*2*0.30)	PZA	0.30	4.00	1.20
2	-	PINTURA AL OLEO	GL	0.005	110.00	0.55
3	-	ESTUCO BEDOYA	KG	0.02	0.60	0.01
4	-	ALAMBRE DE AMARRE	KG	0.02	12.00	0.24
5	-	CLAVOS	KG	0.01	12.00	0.12
6	-	LIENZA PARA ALBAÑIL	RLLO	0.05	5.00	0.25
						
>	D	TOTAL MATERIALES			(A) =	2.37
	B	MANO DE OBRA				1.32
1	-	TOPOGRAFO	hr	0.04	19.00	0.76
2	-	ALARIFE	hr	0.05	11.20	0.56
>	E	SUBTOTAL MANO DE OBRA			(B) =	1.32
	F	Cargas Sociales		55.00% de	(E) =	0.73
	D	Impuesto al Valor Agregado		14.94% de	(E+F) =	0.31
>	G	TOTAL MANO DE OBRA			(E+F+D) =	2.35
	C	EQUIPO, MAQUINARIA Y HERRAMIENTAS				1.28
1	-	EQUIPO TOPOGRAFICO	hr	0.04	32.00	1.28

Anexo 3

Análisis de precio unitarios

En esta imagen se observa los módulos en el que fue dividido la obra. Indicando el monto por cada uno de ellos.

ANALISIS DE PRECIOS UNITARIOS						
PROYECTO: CONST. ENLOSETADO CALLE TUPAC AMARU						
LOCALIZACION: SAN FELIPE DE SEQUE SECTOR 2						
Nº	Descripción	Und.	Cantidad	Unitario	Literal	Parcial
OBRAS PRELIMINARES						
1	INST. DE FAENAS Y COLOC. DE LETRERO OBRAS MENORES (VIAS)	GLB	1.00	2.462,84	Dos Mil Cuatrocientos Sesenta y Dos 84/100 Bs.	2.462,84
2	REPLANTEO Y TRAZADO DE VIAS	ML	123,23	7,39	Siete 39/100 Bs.	910,67
3	EXCAVACION CON TOPADORA	M3	114,54	19,67	Diecinueve 67/100 Bs.	2.253,00
4	EXCAVACION DE 0 - 1,00 M S/AGOTAMIENTO TERRENO SEMIDURO	M3	4,14	51,09	Cincuenta y Uno 09/100 Bs.	211,51
5	CORDON DE ACERA 20X40 CM	ML	41,43	157,49	Ciento Cincuenta y Siete 49/100 Bs.	6.524,81
6	RELLENO Y COMPACTADO MATERIAL COMUN C/COMP. TIPO SALTARIN	M3	2,07	61,40	Sesenta y Uno 40/100 Bs.	127,10
OBRAS DE ENLOSETADO						
7	PERFILADO Y COMPACTADO DE SUBRASANTE	M2	620,17	10,17	Diez 17/100 Bs.	6.307,13
8	LOSETAS ONDULADAS E=10 CM (PROV TRANSP. Y COLOC.)	M2	556,56	154,10	Ciento Cincuenta y Cuatro 10/100 Bs.	85.785,90
9	CORDON DE SUJECION 20X30 CM	ML	11,24	129,22	Ciento Treinta y Nueve 22/100 Bs.	1.454,83
10	CUNETAS DE HORMIGON SIMPLE E= 5 CM, ANCHO = 25 CM	ML	245,94	56,82	Cincuenta y Seis 82/100 Bs.	13.951,24
11	RETIRO DE MATERIAL EXCEDENTE C/CARGUID	M3	139,52	28,48	Veintiocho 48/100 Bs.	3.973,53
12	LIMPIEZA GENERAL DE OBRA	GLB	1.00	1.155,47	Un Mil Ciento Cincuenta y Cinco 47/100 Bs.	1.155,47
Total presupuesto:						125.082,72
Son: Ciento Veinticinco Mil Ciento Ocho con 13/100 Bolivianos						