

UNIVERSIDAD BOLIVIANA DE INFORMÁTICA

CARRERA DE INGENIERÍA DE SISTEMAS



UNIVERSIDAD BOLIVIANA
DE INFORMÁTICA

INFORME

Investigación

UML

POR:

Univ. Emersson Camacho Cori

ASIGNATURA:

Taller de Sistemas

DOCENTE:

Ing. Wilfredo Mendoza Murillo

La Paz-Bolivia

2024

1.	Caratula.....	Pg. 1
2.	Índice.....	Pg. 2
3	Definición de UML.....	Pg. 3
4	Ventajas de UML.....	Pg. 4
	4.1 Dinamismo.....	Pg. 4
	4.2 Claridad.....	Pg. 4
	4.3Facilidad.....	Pg. 4
	4.4 Estandarización	Pg. 4
	4.5 Versatilidad.....	Pg. 4
5-	Diagramas de comportamientos.....	Pg. 5
	5.1 Diagramas de Estado.....	Pg. 5
	5.2 Diagramas de Actividad.....	Pg. 6
	5.3 Diagramas de Caso de Uso.....	Pg. 7
	5.4 Diagramas de Secuencia.....	Pg. 8
	5.5 Diagramas de Comunicación.....	Pg. 10
6-	Diagramas estructurales.....	Pg. 11
	6.1 Diagramas de clase.....	Pg. 11
	6.2 Diagramas de componentes.....	Pg. 12
	6.3 Diagramas de implementación.....	Pg. 13
	6.4 Diagramas de paquete.....	Pg. 14
7-	Diagramas actuales del proyecto.....	Pg. 16
	7.1 Diagrama de clases.....	Pg. 16
	7.2 Diagrama de casos de uso.....	Pg. 17
8-	Conclusión.....	Pg. 19
9-	Bibliografía.....	Pg. 20

3. Definición de UML:

El UML (Unified Modeling Language), es un lenguaje de modelado visual de software, importante para el desarrollo y la arquitectura de software y sistemas.

Y fue pensado y creado como una lengua franca o lengua universal para los desarrolladores, un lenguaje para simplificar y unificar lenguajes de modelación.

Este lenguaje se creó para ser estandarizado para facilitar y mejorar la comunicación y comprensión entre desarrolladores, diseñadores, analistas y cualquier equipo relacionado al desarrollo de software.

Gracias a eso es un modelado gráfico para diagramar, diseñar, especificar, construir, modificar y documentar sistemas.

También fue creado para ser un lenguaje para comunicar sistemas complejos a clientes y stakeholders sin un background técnico, ya que los lenguajes de modelación no utilizan código.

Antes no existía un lenguaje exitosamente estandarizado para el modelado de software. Al contrario, existía una gran variedad de modelos de todo tipo, lo que hacía casi imposible la colaboración inmediata entre equipos de desarrollo.

UML fue presentado en 1996 a la OMG, tras los esfuerzos en conjunto de Ivar Jacobson, Grady Booch y James Rumbaugh, tres pioneros del lenguaje de modelación.

Con UML se pudo establecer un conjunto de estándares y protocolos para el modelado de sistemas, actualmente es un lenguaje universal para el modelado de código, y uno de los más extendidos en todo el mundo.

4. Ventajas de UML

4.1 Dinamismo:

Ya que es un lenguaje basado en notaciones y diagramas, es perfecto para el desarrollo y la simplificación de todo tipo de sistemas de software, ya sean simples o complejos.

4.2 Claridad:

Gracias a una notación ya establecida es, simple y unificada, también es muy eficaz para los desarrolladores, en este lenguaje no queda espacio para la ambigüedad, la vaguedad de conceptos ni la confusión, gracias a ello resulta más fácil de entender y más útil al momento de coordinarse con otros equipos.

4.3Facilidad:

Este modelo se basa en diagramas comprensibles para el usuario común, similares a esquemas o mapas conceptuales.

4.4 Estandarización:

UML es el lenguaje de modelaje de código más común en la actualidad, por ende puedes contar con una amplia gama de recursos para manejarlo, así como una gran comunidad de desarrolladores.

4.5 Versatilidad:

este modelo no se limita al modelado visual de software, sino a la creación e implementación de metodología ágiles, así como a la planificación y análisis de riesgos al implementar sistemas.

5. Diagramas de comportamientos

Los Diagramas de Comportamientos en UML se enfocan en capturar la funcionalidad dinámica de un sistema. Representan cómo el sistema funciona según su flujo de control y manejo de datos.

Existen varios subtipos de diagramas de comportamientos, cada uno para representar distintos aspectos dinámicos:

5.1 Diagramas de Estado

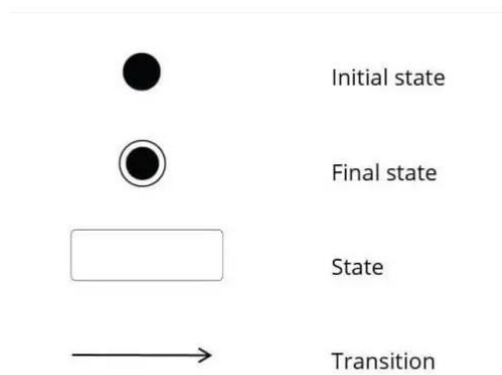
Un diagrama de estado UML (diagrama de transición de estados o diagrama de máquina de estados) muestra los estados por los que pasa una máquina de estados finitos, un modelo de comportamiento que consiste en acciones y estados o transiciones a otros estados.

El diagrama proporciona un estado inicial y uno final, así como al menos un estado intermedio para cada objeto.

El diagrama de estado permite, representar todos los estados que existen en cualquier sistema, subsistema o componentes o clases del mismo.

Figura 1

Elementos



Nota: Elementos que componen un diagrama de estados

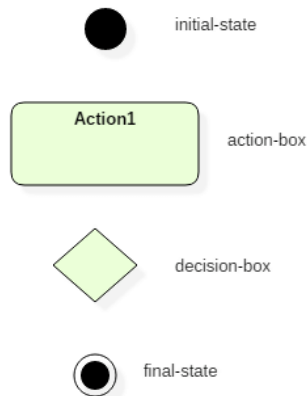
5.2 Diagramas de Actividad:

Un diagrama de actividades representa el flujo de lo que sucede en el sistema en lugar de un estado estático. Visualiza el comportamiento dinámico de un sistema, mostrando el flujo de una actividad a otra.

Cada componente de un diagrama de actividades se llama elemento. Una actividad es el elemento de más alto nivel en el diagrama, generalmente un diagrama de actividades se centre en las partes constituyentes de una sola actividad.

Figura 2

Elementos



Nota: Notación de un diagrama de Actividades

5.3 Diagramas de Caso de Uso:

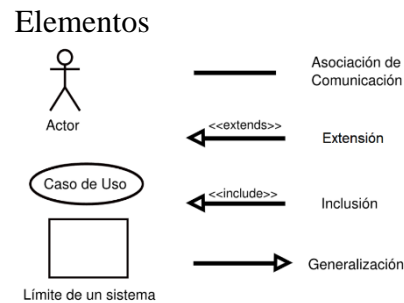
describe el comportamiento que tiene el software en respuesta a eventos realizados por actores externos. También, describe acontecimientos iniciados por el sistema hacia los usuarios.

Los casos de uso son una de las técnicas para especificar los requerimientos funcionales del software. Sólo describen lo que el software debe hacer y para quien. No

entran en detalles sobre como el software será implementado, cuales componentes tendrá o la interacción entre ellos.

Los casos de uso están compuestos de 3 elementos: Actores, casos de uso y relaciones. Asimismo, una especificación de requerimientos funcionales

Figura 3



Nota: Notación del diagrama de casos de uso

5.4 Diagramas de Secuencia:

Es un tipo de diagrama utilizado en ingeniería de software, específicamente en el modelado orientado a objetos, para describir cómo interactúan los diferentes objetos o componentes de un sistema a lo largo del tiempo.

Muestra las interacciones entre los objetos en términos de mensajes que se envían entre ellos, y su secuencia a lo largo del tiempo. Se representa en dos dimensiones: el eje vertical muestra el tiempo, mientras que el eje horizontal muestra los objetos participantes. A través de líneas de vida y mensajes, se especifica el orden en que ocurren las interacciones.

Componentes principales de un diagrama de secuencia:

Objetos o actores: Representados por rectángulos en la parte superior con el nombre del objeto o actor.

Línea de vida: Una línea vertical que desciende desde el objeto, que representa su existencia durante la interacción.

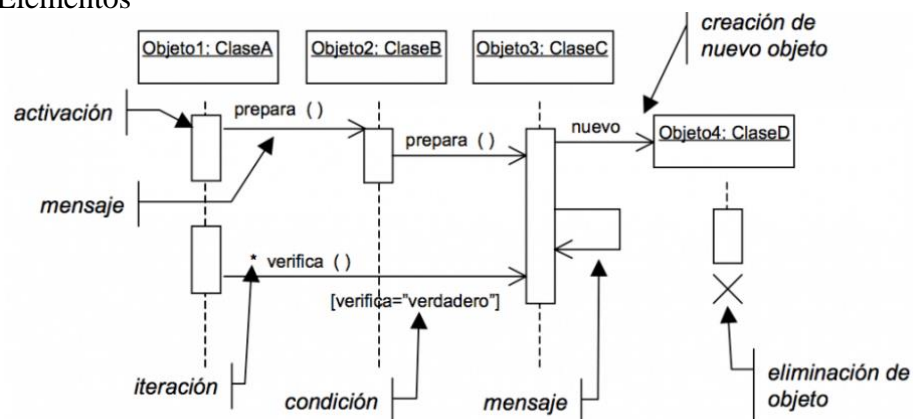
Mensajes: Flechas horizontales que van de un objeto a otro, representando la comunicación o interacción entre ellos.

Activaciones: Bloques rectangulares a lo largo de la línea de vida que muestran cuándo un objeto está ejecutando una acción.

Respuestas: Flechas discontinuas que muestran la respuesta o devolución de información tras el envío de un mensaje.

Figura 4

Elementos



Nota: Notación de los elementos del diagrama de secuencia

5.5 Diagramas de Comunicación:

Un diagrama de comunicación modela las interacciones entre objetos o partes en términos de mensajes en secuencia.

Los diagramas de comunicación representan una combinación de información tomada desde el diagrama de clases, secuencia, y diagrama de casos de uso describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

Los diagramas de comunicación y de secuencia describen información similar, y con ciertas transformaciones, pueden ser transformados unos en otros sin dificultad.

Para mantener el orden de los mensajes en un diagrama de comunicación, los mensajes son etiquetados con un número cronológico y colocados cerca del enlace por el cual se desplaza el mensaje. Leer un diagrama de comunicación conlleva comenzar en el mensaje 1.0, y seguir los mensajes desde un objeto hasta el siguiente, sucesivamente.

Figura 5

Elementos y notacion



Nota: Notación de los elementos de un diagrama de comunicación

6. Diagramas estructurales

los Diagramas Estructurales se centran en la representación de los elementos estáticos que conforman un sistema. Estas estructuras organizan y definen cómo se construye el sistema en términos de componentes interconectados.

Los subtipos de diagramas estructurales incluyen:

6.1 Diagramas de clase:

El diagrama de clases es una herramienta para comunicar el diseño de un programa orientado a objetos, permitiendo modelar las relaciones entre las entidades. En UML, una clase es representada por un rectángulo que posee tres divisiones: Nombre de la clase, atributos que tiene y mensajes que entiende.

En el primer cuadro anotamos el nombre de la clase

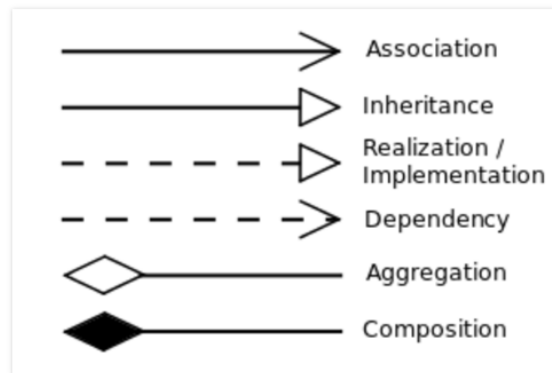
En la segunda parte van los atributos (o variables de instancia, las variables de clase van en subrayado).

En el último cuadro escribimos las operaciones (qué mensajes que puede entender). No confundir con los métodos que es cómo lo resuelve cada objeto. Lo importante no es documentar todos los mensajes de un objeto, sino sólo los más relevantes. Así quedan fuera los getters, setters, métodos privados (o auxiliares) y aquellos que uno considere menores.

Detallan la composición de las clases, incluyendo sus atributos y métodos, así como las relaciones entre ellas.

Figura 6

Elementos y Notación



Nota: Elementos para relacionar clases en un diagrama de clases









6.2 Diagramas de componentes

En el diagrama de muestra cómo se relacionan los componentes entre sí dentro de un sistema más grande.

Un componente del sistema es un módulo que forma parte de un sistema de hardware y software más grande. Tiene sus propias entradas y salidas e interfaces específicas con otros componentes del sistema.

Figura 7

Elementos y Notación

Elemento	Símbolo/notación	Explicación
Componente (<i>component</i>)		Símbolo para representar los módulos de un sistema (la interacción y la comunicación tienen lugar a través de interfaces).
Paquete		Un paquete combina varios elementos del sistema (por ejemplo, clases, componentes o interfaces) en un grupo.
Artefacto		Los artefactos son unidades físicas de información (por ejemplo, código fuente, archivos .exe, scripts o documentos) que se generan en el proceso de desarrollo o el tiempo de ejecución de un sistema o son necesarios para estos.
Interfaz ofrecida		Símbolo para una o más interfaces claramente definidas que proporcionan funciones, servicios o datos al mundo exterior (el semicírculo abierto también se denomina enchufe o <i>socket</i>).
Interfaz requerida		Símbolo de una interfaz necesaria para recibir funciones, servicios o datos del exterior (la notación del círculo con palo también se denomina <i>lollipop</i> o <i>piruleta</i>).
Puerto		Este símbolo indica un punto de interacción independiente entre un componente y su entorno.
Relación		Las líneas actúan como conectores e indican las relaciones entre los componentes.
Relación de dependencia		Conector especial para expresar una relación de dependencia entre los componentes del sistema (no siempre se indica).

Nota: Elementos de un diagrama de componentes (IONOS, 2020)

6.3 Diagramas de implementación

El diagrama de implementación es diagrama que especifica el hardware físico en el que se ejecutará el sistema de software. También determina cómo se implementa el software en el hardware subyacente.

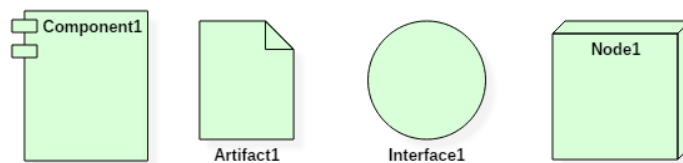
Asigna piezas de software de un sistema al dispositivo que lo ejecutará.

También mapea la arquitectura de software creada en el diseño con la arquitectura del sistema físico que la ejecuta.

En sistemas distribuidos, modela la distribución del software entre los nodos físicos.

Figura 8

Elementos y Notación



Nota: Elementos y notación de un diagrama de componentes

6.4 Diagramas de paquete:

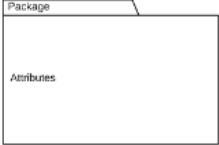

Un diagrama de paquete representa las dependencias entre los paquetes que componen un modelo, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones.

Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes.

Figura 9

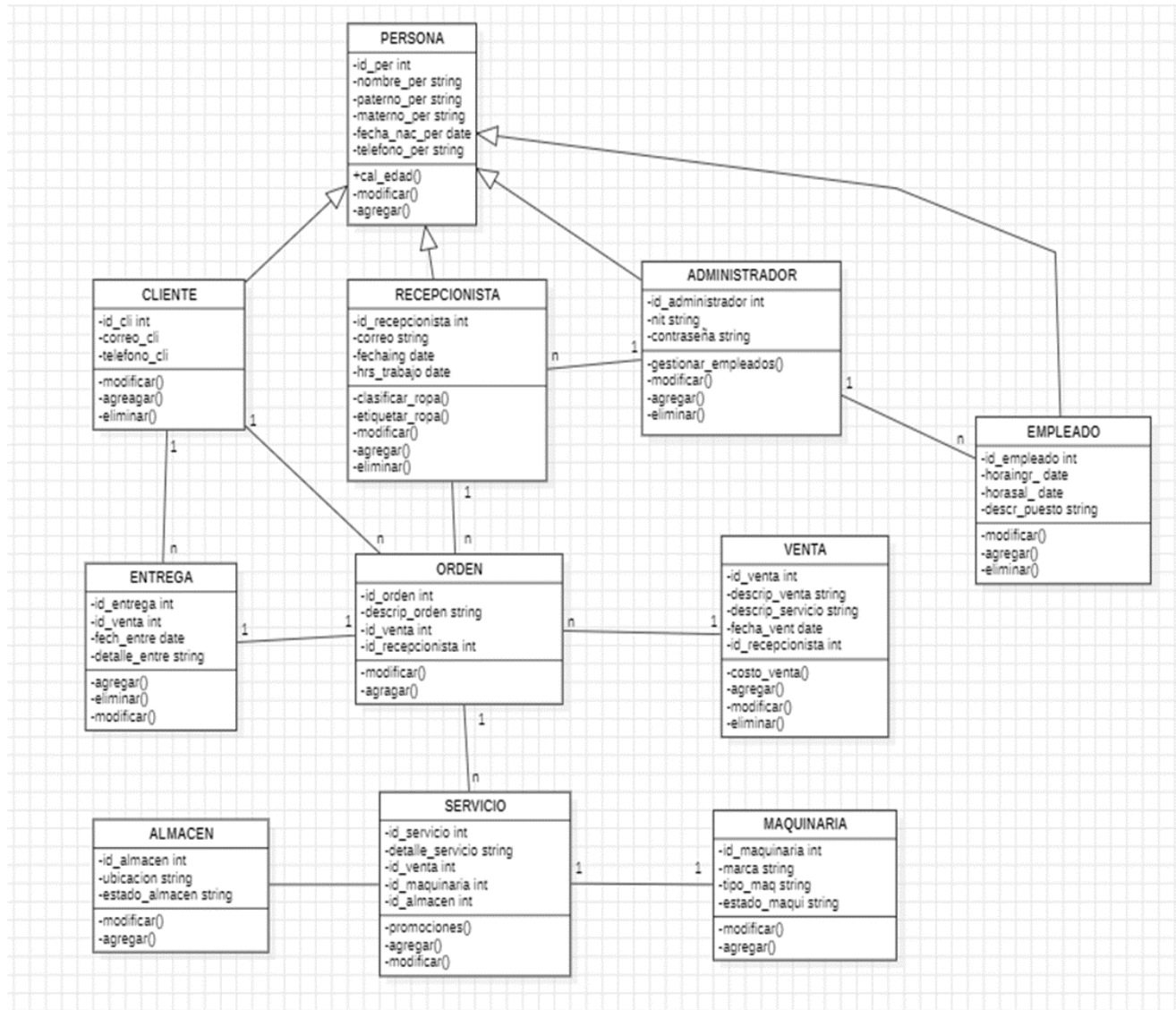
Elementos y Notación

Imagen de símbolo	Nombre del símbolo	Descripción
	Paquete	Agrupar elementos comunes basados en datos, comportamientos o interacciones de los usuarios.
	Dependencia	Muestra la relación entre un elemento (paquete, elemento nombrado, etc.) y otro.

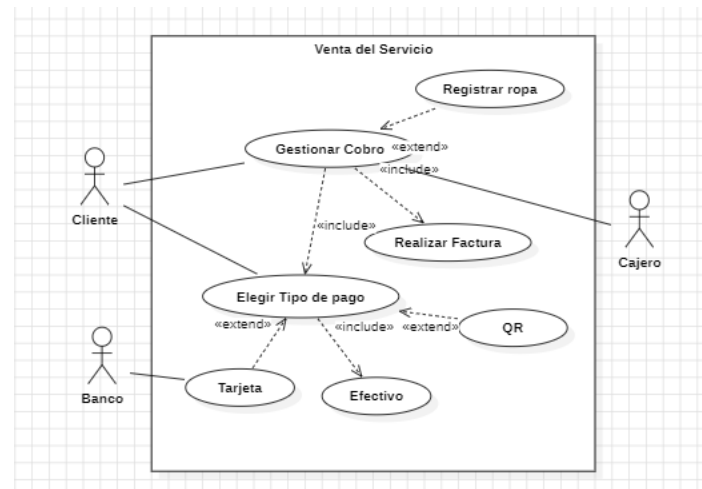
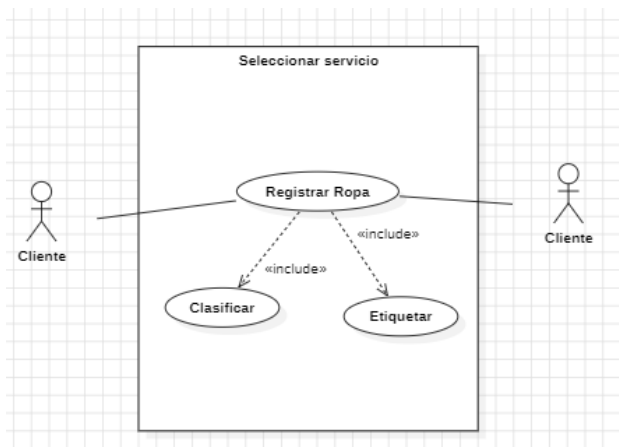
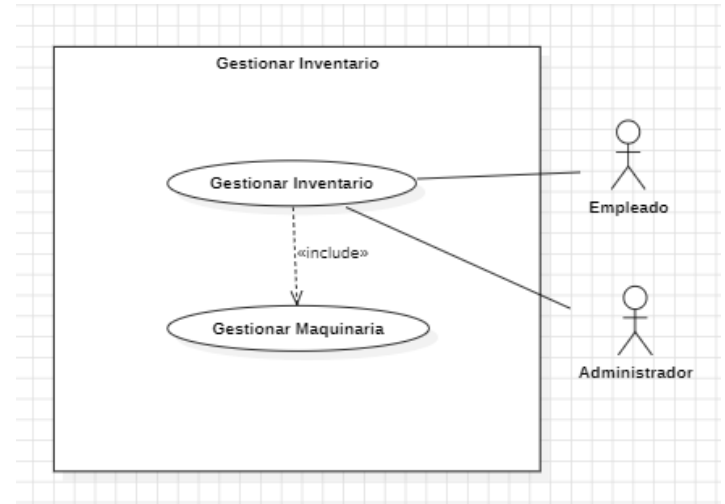
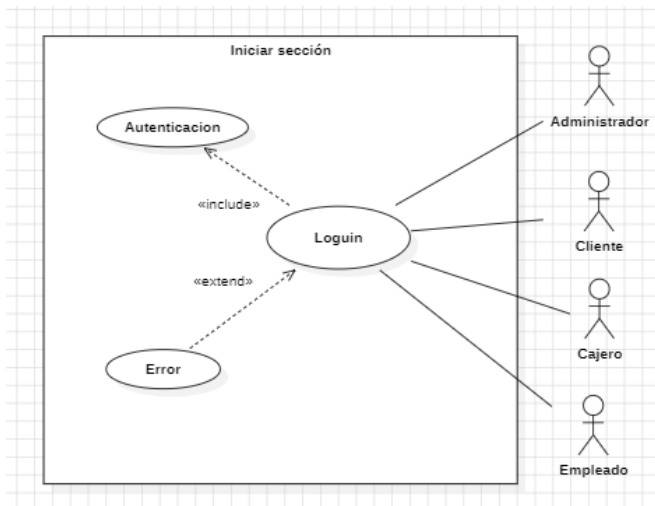
Nota: Elementos del diagrama de componentes

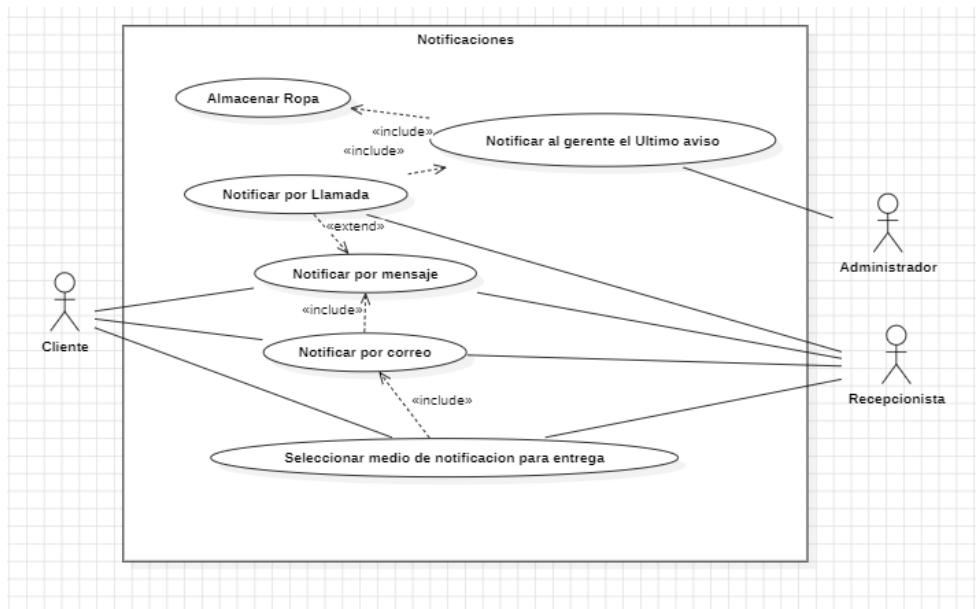
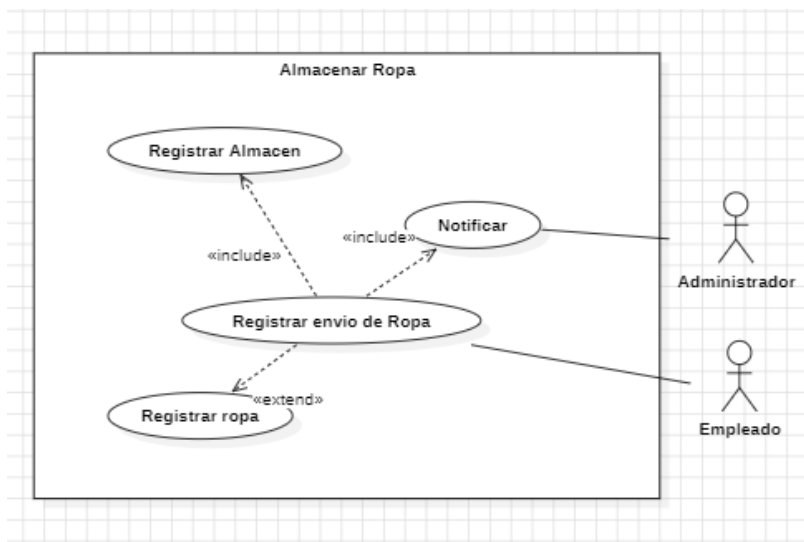
7. Diagramas actuales del proyecto:

7.1 Diagrama de clases:



7.2 Diagrama de casos de uso:





8. Conclusión:

El lenguaje Unificado de Modelado o “UML” nos sirve para documentar todo el sistema en desarrollo, desde la fase inicial hasta la fase final , fue inventado ya que antiguamente existía solo el código “espagueti”, que quiere decir no había una estructura bien definida y cada uno hacia lo que mejor le parecía, también al momento de incorporar nuevos miembros al equipo de trabajo al no tener un modelo claro y bien definido eran más complicado que se adapten a lo que estamos haciendo, el UML gracias a su gran variedad de diagramas también nos ayudan a desglosar el sistema a desarrollar ya sea un sistema simple o complejo, también gracias al UML la colaboración entre equipos es más eficiente yaqué se comunican con un lenguaje en común estandarizado, tener en cuenta que el UML no solo se puede usar dentro de sistemas también en otras áreas como el sector financiero y telecomunicaciones entre otras.

9. Bibliografía

Acevedo, F. (15 de Febrero de 2024). *nuvaweb*. Obtenido de <https://nuvaweb.com>

Brutti, F. (03 de Mayo de 2023). *ThePower*. Obtenido de <https://global.thepower.education>

IONOS. (07 de Agosto de 2020). *IONO Digital Guide*. Obtenido de <https://www.ionos.com/es-us/digitalguide/>