**Documento de Diseño de Software**

**IEEE 1471-2000**

**Control del documento**

**Proyecto**

Sistema Web para la Gestión de Accesos y Aforo por COVID-19 “CovSafe”

**Título**

Diseño de software – [V1.0 al 11 de noviembre de 2020]

**Generado por**

Estudiante de ingeniería de sistemas y computación – [Isabela Acevedo García]

Estudiante de ingeniería de sistemas y computación – [Iván Darío Bedoya Suárez]

Estudiante de ingeniería de sistemas y computación – [Victor Manuel Toro Cedeño]

# Introducción

## Propósito

El siguiente documento proporciona una descripción de la arquitectura del sistema que se pretende implementar para gestionar el sistema web para la gestión de accesos y aforo por COVID-19. Utilizando diferentes tipos de vistas para representar un mismo sistema y las funcionalidades que hay dentro del mismo.

## Alcance

El presente documento contiene el proyecto CovSafe representado con diferentes diagramas realizados a partir del análisis de los requisitos levantados para la gestión de acceso y aforo por COVID-19.

El documento se organizó de tal manera que primero se presenta la elección de la arquitectura del software y el gestor de bases de datos, dependiendo algunos atributos de calidad. Después, se presenta la arquitectura del software vistos desde distintos diagramas.

Cada uno de los modelos fueron implementados en la herramienta de modelado UML (Lenguaje de modelado unificado).

## Definiciones, acrónimos y abreviaciones

|  |  |
| --- | --- |
| ***Nombre*** | ***Descripción*** |
| **Arquitectura de software** | Conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del software en tiempo de ejecución. |
| **Atributos de calidad** | Características no funcionales que se consideran deseables en un sistema de software. |
| **DAS** | Documento de arquitectura de software |
| **Escalabilidad** | Es la capacidad de un sistema para trabajar con diferentes cantidades de trabajo, como cambios en el volumen de datos o flujo de pedidos. |

# Referencias

[1] G. A. Salazar Garzón, ”13. Diseño de Software”, *Aulavirtualdecc*, 2020. [En Línea]. Disponible en: <https://aulavirtualdecc.javerianacali.edu.co/pluginfile.php?file=%2F4359%2Fmod_resource%2Fcontent%2F1%2F13.%20Dise%C3%B1o%20de%20Software.pdf> [Accedido: 10-oct-2020]

[2] G. A. Salazar Garzón, “14. Diseño de Software II”, *Aulavirtualdecc*, 2020. [En Línea]. Disponible en: <https://aulavirtualdecc.javerianacali.edu.co/pluginfile.php?file=%2F4503%2Fmod_resource%2Fcontent%2F1%2F14.%20Dise%C3%B1o%20de%20Software%20II.pdf> [Accedido: 10-oct-2020]

[3] “Atributos de calidad”, *uqbar*, 2020. [En Línea]. Disponible en: <http://wiki.uqbar.org/wiki/articles/atributos-de-calidad.html#:~:text=Atributos%20de%20calidad%20> [Accedido: 10-oct-2020]

[4] R. Marín, “Los gestores de bases de datos más usados en la actualidad.” , *revistadigital*, 2019. [En Línea]. Disponible en: <https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/> [Accedido: 10-oct-2020]

[4] L. Peris, “¿Qué base de datos escoger?”, *Luis Peris*, 2017. [En Línea]. Disponible en: <https://luisperis.com/base-datos-escoger/> [Accedido: 10-oct-2020]

# Representación Arquitectónica

Para escoger las arquitecturas candidatas para el proyecto de CovSafe primeramente se tuvieron que tener en cuenta los atributos de calidad que deben estar en el sistema, los cuales son disponibilidad, rendimiento, testeabilidad. escalabilidad y seguridad. La disponibilidad , ya que uno de los requisitos del sistema es que deberá atender solicitudes el 99% del tiempo, es decir, debe estar disponible el 99% del tiempo.Rendimiento, otro de los requisitos es que cuando un usuario haga una solicitud su tiempo de espera máximo debe ser de 2 segundos. Testeabilidad , debido a que un error en la comunicación puede generar que un usuario registrado con covid19 por una entidad de salud se le autorice el ingreso a un establecimiento, este error puede poner en peligro a muchas personas por eso se requiere que se puedan hacer pruebas globales en el sistema . Escalabilidad, como la aplicación se quiere diseñar para una ciudad completa primeramente , el número de usuarios puede ir incrementando, por lo que se requiere que el sistema puede adaptarse a esa posibilidad .Y por último la seguridad, al manejar información personal de los usuarios , establecimientos y entidades de salud, otro de sus requisitos es que el sistema tenga métodos de autenticación ,encriptación de contraseñas,etc,es decir , que garantice la seguridad de la información.

La primera arquitectura que se tuvo en cuenta fue la “orientada a eventos” , ya que esta por su manera de ser implementada permite tener eficiencia en las solicitudes de los usuarios , el problema de esta arquitectura es que a pesar de que nos brinde esta ventaja ,también trae consigo algunas desventajas, como lo es la testabilidad y la inconsistencia de los datos. En cuanto a la testabilidad dado que su funcionamiento como su nombre lo indica es orientado a eventos, no hay manera de hacer una prueba sencilla esperando el resultado si el evento fue terminado, por lo que habría que implementar eventos más rebuscados con el que se puede comprobar el correcto funcionamiento de un proceso, y en cuanto a la inconsistencia que se puede generar en los datos , debido a que si un registro falla , no significa que el resto de comunicaciones se vean afectados , por lo que un usuario registrado con Covid , podría visitar muchos lugares que si el sistema falla dando la información del usuario puede ser peligroso para los otros visitantes del sitio.

Otra de las arquitecturas que se tuvo en cuenta fue “Microservicios” , debido a que el modelo y la naturaleza de la arquitectura en general permite una buena escalabilidad y testeabilidad . En cuanto a la escalabilidad debido al desarrollo por módulos o por servicios que tiene la arquitectura, en caso de querer agregar más funcionalidades al sistema solo se tiene que agregar otro microservicio. Y por último la testabilidad , de nuevo como su desarrollo es por módulos, hace que probar cada componente sea más sencillo y verificar su correcto funcionamiento. Los problemas principales que tiene la arquitectura para empezar es el rendimiento, como van a existir en esta arquitectura varios microservicios, si una solicitud debe pasar por muchos de los componentes puede ser más lento el tiempo de respuesta, y además a pesar de que anteriormente se dijo que una de sus ventajas era las testeabilidad , tambien tiene un inconveniente, y es que al probar el sistema completo se hace algo más complicado debido al número de componentes , es decir, hacer las pruebas globales del sistema se hace más tedioso con esta arquitectura.

Y por último tenemos la arquitectura “por capas” , esta arquitectura por naturaleza ofrece un gran rendimiento y testeabilidad . Como su diseño es por capas (normalmente son 3), la comunicacion y atencion de las solicitudes suele ser rápida dependiendo de la implementación , por que solo debe viajar por 3 capas en general, en las cuales cada capa se ocupa una sola funcionalidad, por lo que no se genera el problema que existe en “microservicios” de tener que viajar la solicitud por varios componentes. La testeabilidad tanto global como por componentes es fácil de realizar debido a que si se quiere probar las funcionalidad de cada capa, basta con realizar pruebas en cada una de ellas , y si se quieren hacer pruebas globales es más sencillo , debido a que cada solicitud , si necesita algo de la base de datos por ejemplo debe viajar por todas las capas . Los problemas de esta arquitectura principalmente es la escalabilidad, debido a que para agregar nuevas funcionalidades se necesita cambiar toda la capa a la que vaya pertenecer el componente, pero como la escalabilidad que se busca en este sistema es más por números de usuarios que de funcionalidades,

Por todas las ventajas y desventajas de las arquitecturas nombradas anteriormente decidimos que la mejor arquitectura para nuestro sistema será “Por capas” ya que además de ser de las más simples de implementar , es la que mayor ventajas no genera en la implementación del proyecto y la que mayor atributos de calidad cumple. Cabe resaltar que otros atributos como la seguridad y la disponibilidad dependen más a la hora de implementar la arquitectura y no de sus naturalezas , por eso no se tuvieron tan en cuenta en esta sustentación.

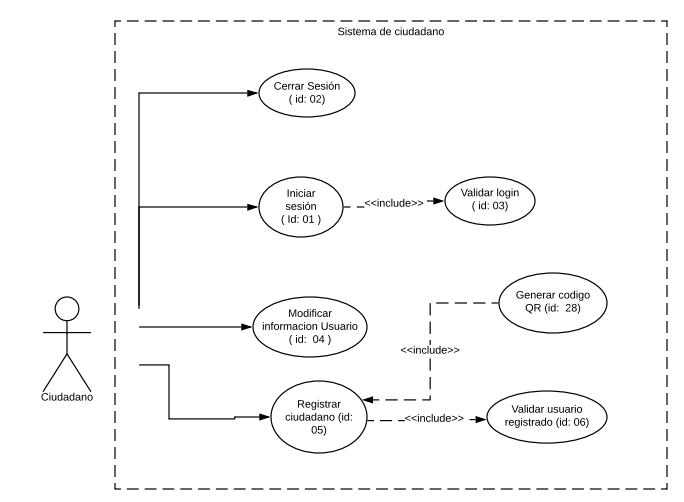
# Objetivos y Restricciones de la arquitectura de software

Los objetivos que se buscan cumplir con la arquitectura de software por capas, es la de tener un buen rendimiento , específicamente que los usuarios no tengan que esperar por tiempos de respuestas mayores a 2 segundos. Otro de los objetivos que se busca es la testeabilidad , que se pueda probar sin mayores complicaciones cada capa y el funcionamiento global del sistema. Además de estos atributos también se quiere garantizar la durabilidad de los datos, por lo que la arquitectura de cierta manera no se debe ver afectada por sí una de las base de datos sufre fallos.

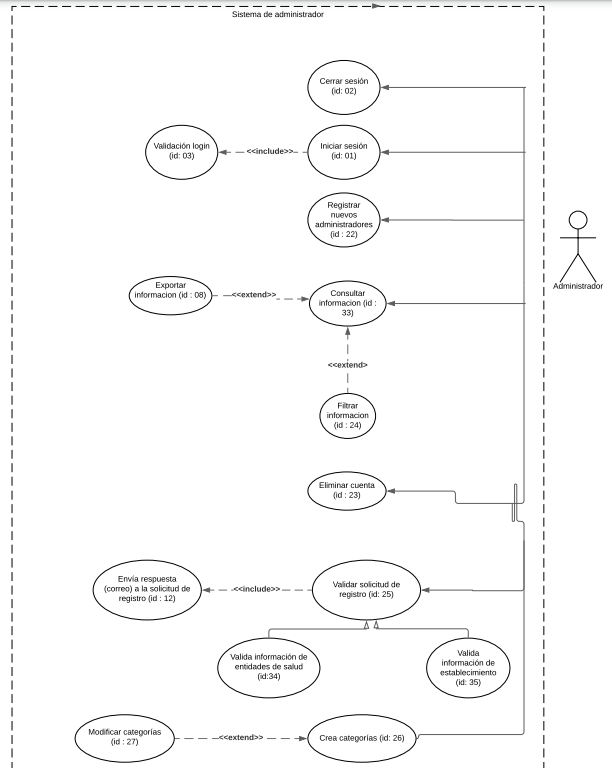
Las restricciones de la arquitectura como tal será la escalabilidad en cuanto a agregar nuevas funciones se refiere, al ser una arquitectura por capas , realizar esta tarea se hace más difícil ya que habría que modificar toda la capa en general.

# Vista de casos de uso

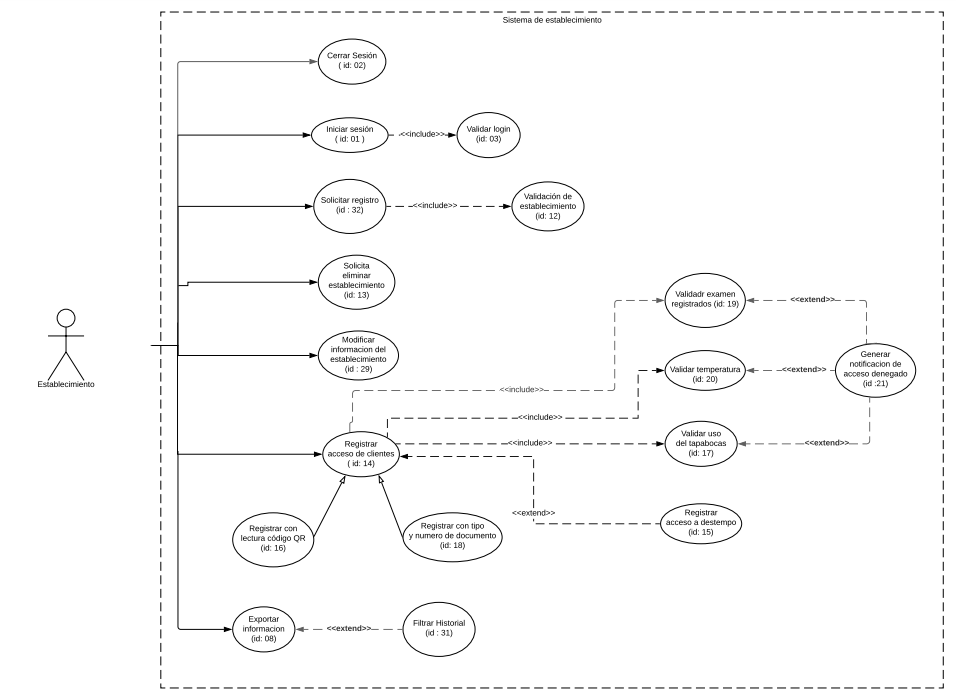
* **CIUDADANO**

****

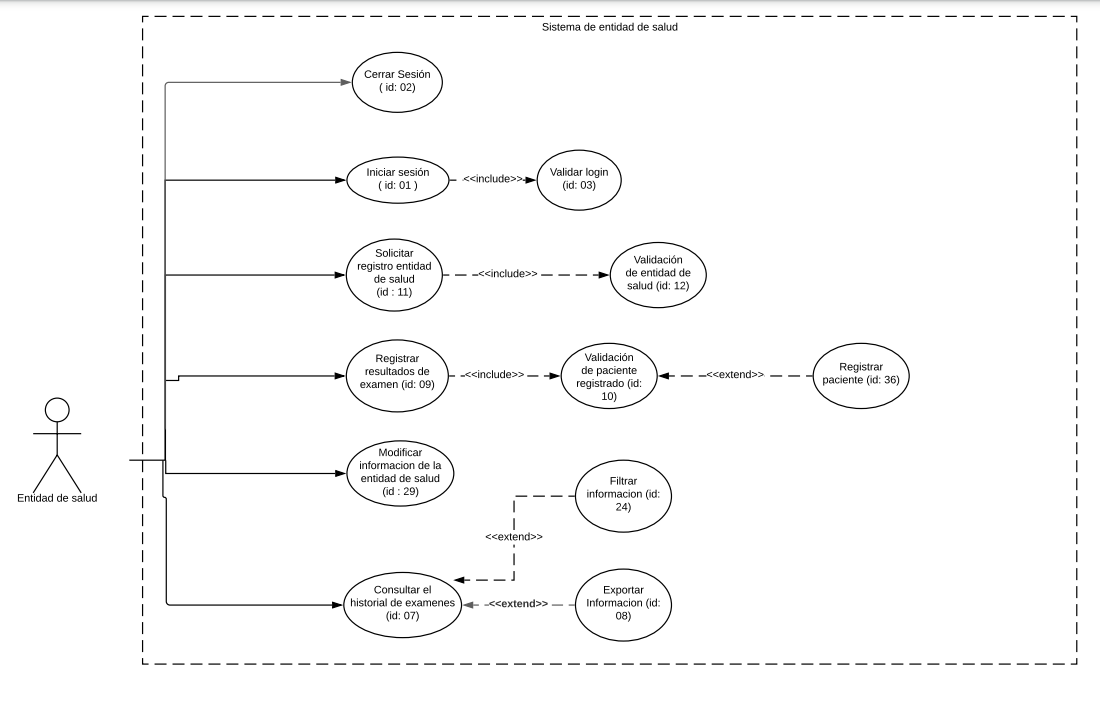
* **ADMINISTRADOR**



* **ESTABLECIMIENTO**

****

* **ENTIDAD DE SALUD**

****

# Vista lógica

* **Modelo de datos**

Para escoger el tipo de bases de datos a utilizar en el sistema se tuvieron en cuenta los atributos de calidad , los tipos de bases de datos candidatas fueron relacional, no relacional, documental y orientada a objetos.

A continuación se mostraran las ventajas y desventajas de los tipos de bases de datos seleccionadas para concluir el por que de la base de datos finalmente escogida.

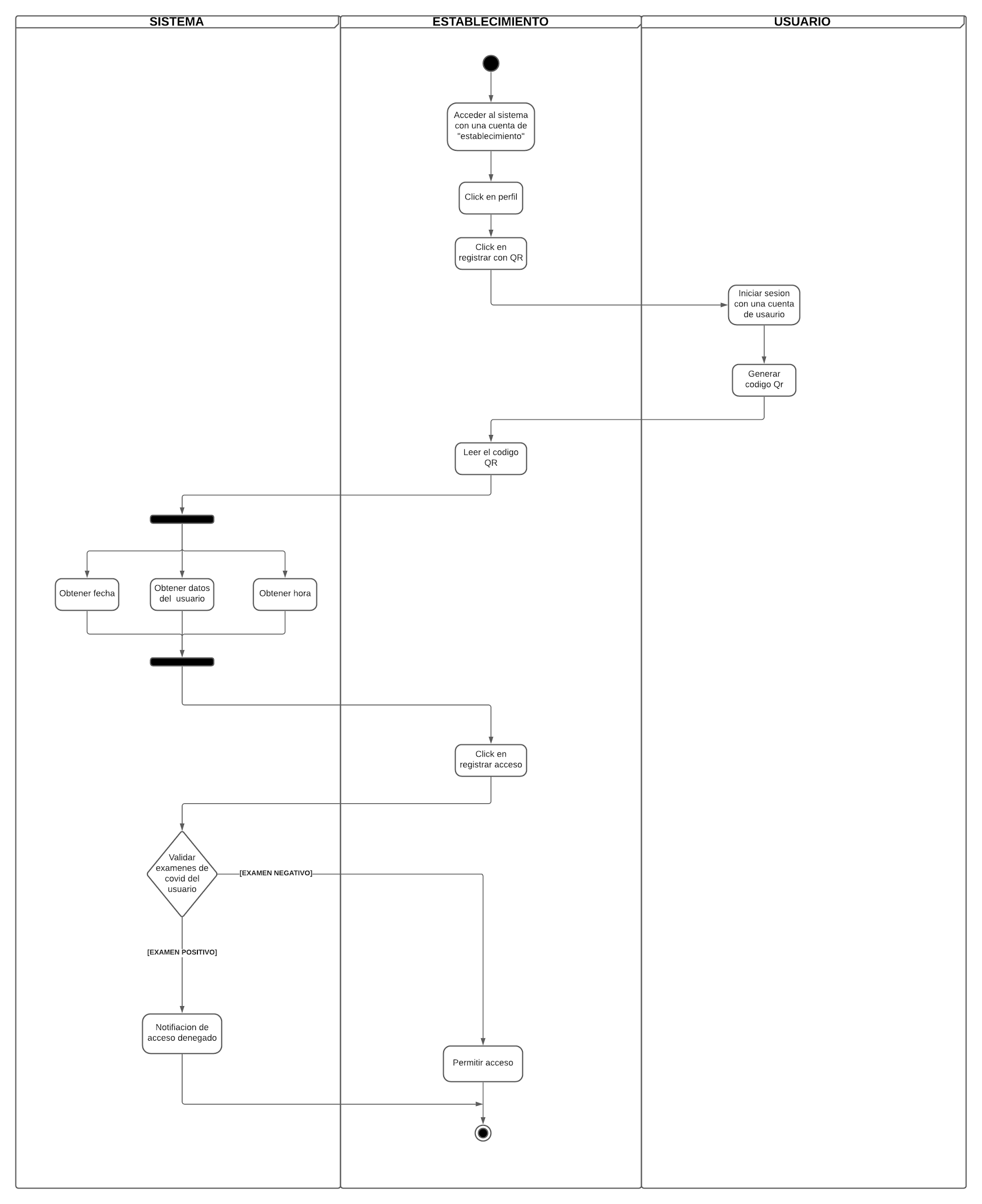
|  |  |  |
| --- | --- | --- |
|  | **VENTAJAS** | **DESVENTAJAS** |
| **RELACIONAL** | -Asegura la consistencia de los datos, es decir que no se repitan datos con el mismo identificador.  -Es del tipo de bases de datos más usado, por lo que tiene un estándar definido, lo que significa que realizar las consultas se hace de manera sencilla. | -Como las tablas de este tipo de base de datos están relacionadas de alguna manera entre sí, hace que sea complicado agregar nuevas tablas o tipos de datos.  -No es escalable, es decir, es difícil hacer que la base de datos soporte un número de usuarios mayor al previsto.  -Cómo garantiza la consistencia de los datos, cumplir estos requisitos hace que el rendimiento del sistema se vea afectado. |
| **NO RELACIONAL** | -Es escalable, no está limitada por un número de datos o por picos de acceso.  -Este tipo de base de datos mejoran el rendimiento de las consultas para que sean más rápidas.  -No se requiere de una gran máquina para soportar este tipo de base de datos.  -Agregar nuevos tipos de datos se realiza de manera sencilla. | -Algunos sistemas no logran garantizar la consistencia de datos.  -No tiene un estándar específico por lo que la documentación puede ser algo más complicada. |
| **DOCUMENTAL** | -Como todos los datos se almacenan en forma de texto hace que el manejo de grandes volúmenes de datos sea más eficiente.  -Permite que el rendimiento del sistema mejore si los datos no están relacionados entre sí,es decir, mejora el tiempo de consultas | -No pueden garantizar las ACID(atomicidad, consistencia, integridad y durabilidad.) de los datos.  -Como no tiene un estándar establecido, entender la documentación de este tipo de base de datos se hace complejo. |
| **ORIENTADA A OBJETOS** | **-**Los conjuntos de datos complejos se pueden almacenar y consultar de manera rápida.  -Ampliabilidad, se pueden agregar nuevos tipos de datos de manera sencilla por lo que este tipo de bases es escalable. | -Al estar los objetos relaciones con punteros , hace que el rendimiento sea mucho menor que otras bases de datos  -No tiene un estándar establecido, por lo que la documentación se hace algo más difícil de entender |

Después de mostrar las ventajas y desventajas de estos tipos de bases de datos , se decidió por usar una base de datos no relacional, ya que sus desventajas no afectan tanto al sistema en general, y a su vez que sus ventajas cumplian con la gran mayoría de los atributos de calidad del sistema como lo son la escalabilidad ,el rendimiento y el límite de usuarios.

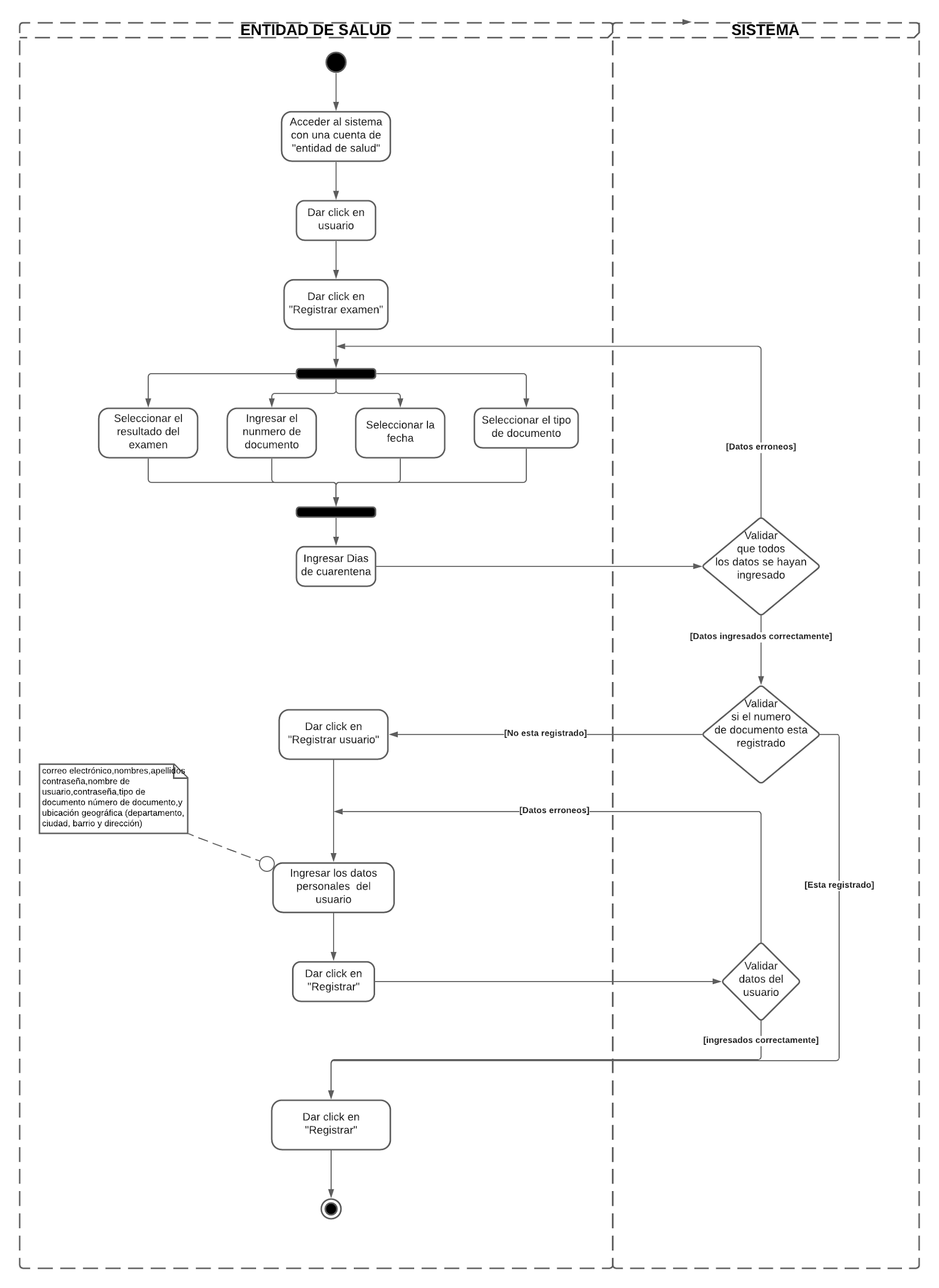
El programa que se usará para diseñar e implementar esta base de datos será MongoDb , ya que es de los más conocidos para las bases de datos no relacionales y ofrece documentación más actualizada que otro gestores de bases de datos.

1. **Vista de procesos**

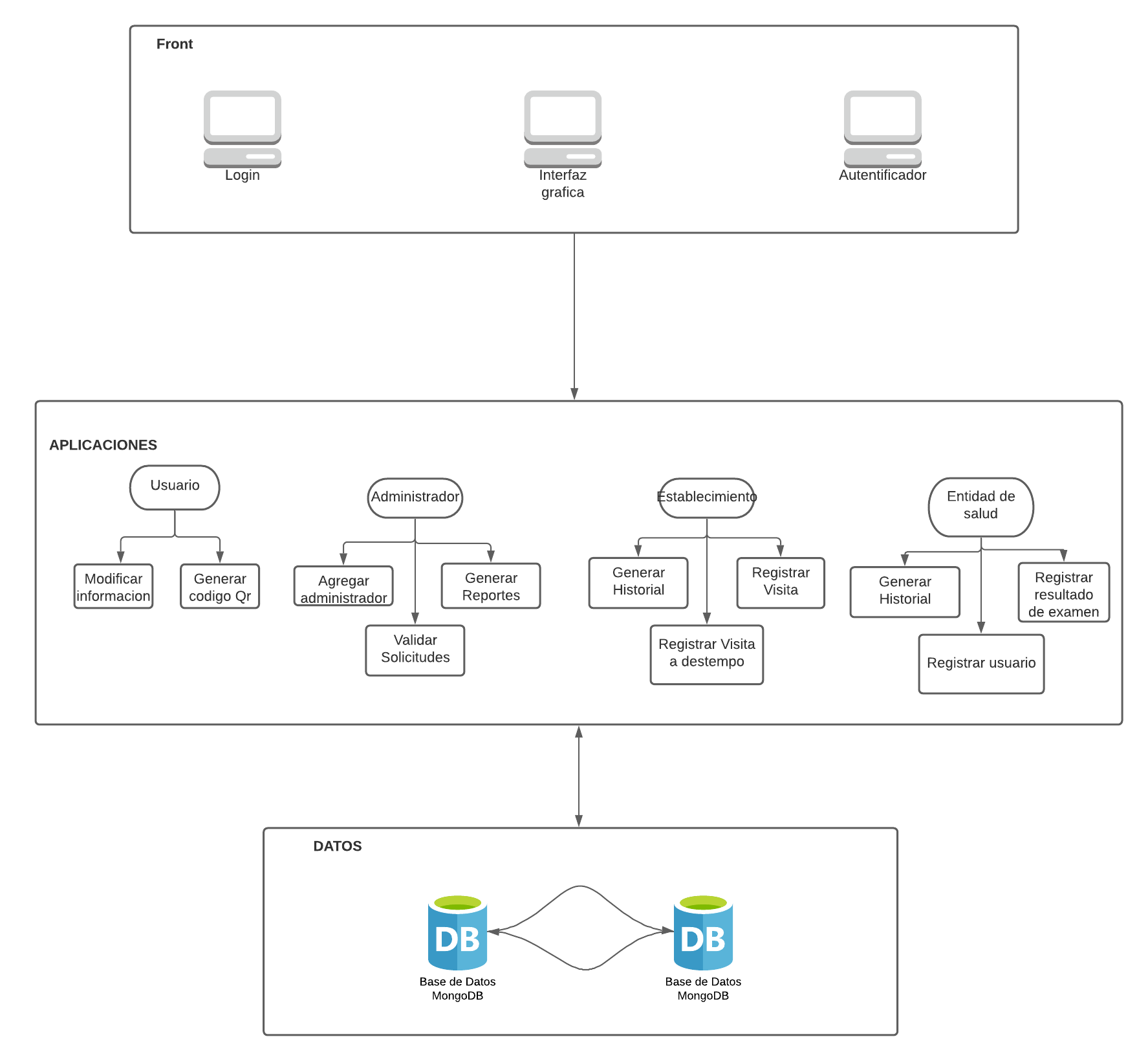
* **Diagrama de actividades 1:**

****

* **Diagrama de actividades 2:**

****

1. **Vista Física**

****

1. **procesos**

Para el despliegue de la aplicación se tendrán las bases de datos en servidores externos , ya que al ser un sistema no muy grande por temas de costos y rendimiento es más rentable hacerlo de esta manera. Para el tema de número de usuarios concurrentes usaremos un balanceador de cargas para que el número de solicitudes en nuestro servidor de aplicaciones no se sobrecargue. Tendremos 3 servidores, el primero donde estará el código fuente del sistema con sus funcionalidades, y también el código flask de python que representa la interfaz gráfica realizada en Html. Para manejar el tema de disponibilidad, usaremos 2 servidores con una base de datos en cada uno, el primer servidor será el principal y el segundo será una copia, ya que si el servidor o la base de datos llega a fallar se tiene el segundo para que evitar tiempos de espera largos mientras se hace el mantenimiento a este mismo.

