Pontificia Universidad Javeriana de Cali

Documento de Diseño de Software

Nutristory App

Gestión de Historias Clínicas de Nutrición

Procesos y Diseño de Ingeniería de Software

Autores:
Ana María García, Andrés Felipe Delgado, Leonardo Saez

Abril 11 del 2020

Índice

1. Introducción

1.1. Propósito

En este documento se especificará el documento de diseño del proyecto de gestión de historias clínicas junto con otros entregables y correcciones de la primer entrega de este proyecto.

Este documento está dirigido a los clientes directos del proyecto, que son el docente Gustavo Salazar y la docente de Nutrición y Dietética.

1.2. Alcance

El alcance de este documento será definir todos los puntos que contiene el documento de diseño del proyecto. Los demás entregables se encontrarán anexados en el portafolio del proyecto.

Este proyecto tiene como objetivo construir un software que permita gestionar historias clínicas de nutrición utilizando técnicas vistas en clase, con el fin

de poner en práctica la aplicación de estas en un proyecto real de tamaño mediano implementando la metodología RUP.
Este software cuyo nombre será "Nutristory App", será una herramienta que permitirá la gestión y manejo de historias clínicas de nutrición, con el principal beneficio de brindarle a los nutricionistas un manejo sistematizado y ordenado de las historias clínicas de los pacientes que consulten con nutrición, facilitando la creación y el uso de estas historias.

1.3. Definiciones, Acrónimos y Abreviaciones

Las definiciones particulares utilizadas en este proyecto ya fueron definidas en el documento SRS. No se hace uso de definiciones nuevas.

1.4. Referencias

[1] "Programación por capas - Wikipedia, la enciclopedia libre." [Online]. Available: https://es.wikipedia.org/wiki/Programaciónporcapas. [Accessed: 02-May-2020].
[2] J. Badgujar, M. Jailia and A. Kumar, "Performance metrics of web crawler in client-server and MVC architecture,"2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, 2015, pp. 393-398.
[3]"Microservicios Wikipedia, la enciclopedia libre." [Online].[Accessed: 02-May-2020].
[4] A. Garcete, "Base de Datos Orientado a Columnas," 2014, 2014.
[5] E. Anjos and M. Zenha-Rela, "A framework for classifying and comparing software architecture tools for quality evaluation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6786 LNCS, no. PART 5, pp. 270–282, 2011, doi: 10.1007/978-3-642-21934-423.
[6] "Base de datos relacional - Wikipedia, la enciclopedia libre." [Online]. Available: https://es.wikipedia.org/wiki/Basededatosrelacional. [Accessed: 03-May-2020].
[7] "Bases de Datos NoSQL — Qué son, marcas, tipos y ventajas." [Online]. Available: https://www.grapheverywhere.com/bases-de-datos-nosql-marcas-tipos-ventajas/. [Accessed: 03-May-2020].

2. Representación Arquitectónica

Teniendo en cuenta los requisitos no funcionales adjuntos en la carpeta Çorrecciones 1er Entrega2 en el archivo "Documentación de requisitos.pdf", se sacaron 3 atributos de calidad del sistema que son los siguientes:

Rendimiento - RNF02
Testeabilidad - RFN05

Seguridad - RNF04

Estos atributos se definirán en la siguiente tabla:

Rendimiento	Testeabilidad	Seguridad
El rendimiento para nuestro proyecto es relevante puesto que el software en un caso real será usado por servicios medicos que por definición deben ser eficaces y eficientes para considerarse de buena calidad, por tanto, el rendimiento del sistema es clave para obtener una mejora considerable en el servicio con respecto a las historias clinicas en formato fisico.	Consideramos este atributo puesto que es muy importante hacer las pruebas necesarias a nuestro software de manera eficiente antes de salir a producción, pues se desea mitigar la probabilidad de que ocurran fallos ya en etapas de liberación de este, ya que un error en el sistema puede derivar en la perdida o incorrecto almacenamiento de información que es vital para que un paciente pueda ser monitoreado debidamente por el especialista.	La seguridad es un atributo de calidad muy importante para el software pues necesitamos que los usuarios puedan ingresar solo a la información que corresponde a su rol, es decir, el sistema debe proteger los datos de accesos no autorizados y permitir el acceso a partes autorizadas. Este atributo adquiere mayor relevancia si tenemos en cuenta que la información que se va a almacenar en el sistema corresponde al estado de salud de una persona, por lo cual si dicha información se corrompe o roba por un tercero puede tener graves consecuencias.

Figura 1: Atributos de Calidad del Sistema

Teniendo en cuenta estos atributos de calidad, se realizó un proceso de selección de las arquitecturas más acordes al tipo de proyecto que se debe realizar y a los atributos de calidad. Las arquitecturas candidatas son las siguientes:

Arquitectura por capas: "La programación por capas es un modelo de desarrollo software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que componen un sistema software o también una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos. De esta forma, por ejemplo, es sencillo y mantenible crear diferentes interfaces sobre un mismo sistema sin requerir cambio alguno en la capa de datos o lógica."[1]

Arquitectura Cliente servidor: "La arquitectura del servidor del cliente distribuye varias tareas entre el cliente y el servidor. En este, los datos comunes para el usuario son almacenados y procesados por el servidor y luego el cliente puede acceder a los datos. Los diferentes clientes realizan solicitudes al servidor. Luego, el servidor procesa la solicitud del cliente y proporciona los resultados deseados al cliente. El servidor actúa como base de datos."[2]

Arquitectura de microservicios: "Microservicios es una técnica de desarrollo de software, una variante del estilo estructural de la arquitectura orientada a servicios (SOA), que organiza una aplicación como una colección de servicios acoplados libremente. En una arquitectura de microservicios, los servicios son

específicos y los protocolos son livianos."[3]

A continuación se muestran las cualidades que tiene cada arquitectura candidata con respecto a cada uno de los atributos de calidad mencionados anteriormente:

	Atributos de calidad		
	Rendimiento	Testeabilidad	Seguridad
Microservicios	Al separar la aplicación en microservicios el rendimiento se puede ver afectado por la gran cantidad de paso de mensajes que se deben realizar entre servicios	El separar los componentes en microservicios ayuda a aislar estos por lo cual las pruebas se pueden hacer a los componente de manera mas rapida pues no hay dependencias	Componentes aislados permite que la seguridad pueda enfocarse en un solo componente que al ser independiente su seguridad no se vera comprometida
Por capas	El patrón no se presta a aplicaciones de alto rendimiento debido a las ineficiencias de tener que pasar por múltiples capas de la arquitectura para cumplir con una solicitud comercial.	Al ser una arquitectura que distribuye sus funciones por capas, se pueden hacer pruebas sobre cada capa sin que haya dependencia de otras capas, por lo cual es mas eficiente la etapa de pruebas	La división entre capas permite que al no haber dependencia entre estas la seguridad implementada en una capa no se vea afectada por otra
Cliente-Servidor	El rendimiento que se puede obtener en esta arquitectura es bueno aunque este dependera de la capacida del servidor en donde se ejecute.	La testeabilidad en esta arquitectura no es tan simple debido a que todo el sistema esta integrado en una sola aplicación, por lo que los casos de prueba serian muy complejos	Este patron no tiene una seguridad media, pues al ser la aplicación un solo conjunto, si existe una brecha de seguridad, esta compromete a todo el sistema

Figura 2: Arquitecturas Candidatas - Comparación

Arquitectura seleccionada: Arquitectura por capas.

El método de selección de arquitectura se llevó a cabo por medio de la gráfica anterior donde se describieron los atributos de calidad de cada patrón. Después, se realizó una votación en equipo teniendo en cuenta las cualidades previamente descritas evaluando cual de estas tendría mayor peso en el proyecto y de esta forma, se escogió el patrón de arquitectura por capas debido a que dos de tres atributos de calidad(Testeabilidad y seguridad) se ajustan mejor a las necesidades del proyecto.

3. Objetivos y Restricciones de la Arquitectura

La arquitectura seleccionada que fue la arquitectura por capas debe cumplir los siguientes atributos de calidad:

1. Testeabilidad: Facilidad de la realización de pruebas del sistema.
2. Seguridad: No permitir que un usuario cuyo rol no haga parte del sistema (nutricionista o administrador) pueda ingresar a las funcionalidades de la aplicación web.

El objetivo principal de la arquitectura por capas es proporcionar separación entre cada capa. Por lo que cada capa se debe plantear de tal manera que haya muy poco efecto en las otras capas. Actualmente se han creado diferentes técnicas para lograr el desacoplamiento entre capas, y las interfaces son una de las técnicas que más se han usado. Así que en vez de usar implementadores concretos, la comunicación entre capas se lleva a cabo por medio de las interfaces y de esta forma los cambios no afectan al resto.

Restricciones y desventajas:

• El patrón no se presta a aplicaciones de alto rendimiento debido a las ineficiencias de tener que pasar por múltiples capas de la arquitectura para cumplir con una solicitud comercial.
• La escalabilidad puede ser difícil con una arquitectura en capas. Esto está relacionado con el hecho de que muchas aplicaciones en capas tienden a adquirir propiedades monolíticas.
• El desarrollo de aplicaciones intensivas para el usuario puede llevar más tiempo si las capas evitan el uso de componentes de la interfaz de usuario que interactúan directamente con la base de datos.
• Una aplicación en capas es más difícil de evolucionar, ya que los cambios en los requisitos a menudo tocarán todas las capas.
• La arquitectura en capas está muy centrada en la base de datos. Como se mencionó anteriormente, dado que todo fluye hacia abajo, a menudo es la base de datos la última capa. Los críticos de esta arquitectura señalan que su aplicación no se trata de almacenar datos; se trata de resolver un problema de negocios.

4. Vista de los Casos de Uso

La documentación de los casos de uso del sistema se encuentra adjunta la carpeta Çorrecciones 1er Entrega", en el archivo "Documentacion casos de uso.pdf"

En la misma carpeta, El diagrama de casos de uso se dividió entre un diagrama de casos de uso del nutricionista y otro del administrados, estos se encuentran en los archivos "Diagrama casos de uso-nutricionistaz "Diagrama casos de uso-adminrespectivamente.

5. Vista Lógica

5.1. Arquitectura

Los diagramas de clases se encuentran adjuntos en la carpeta .ªrquitectura por capas", en el archivo "Diagrama de clases.pdf".
En la misma carpeta se encuentra los archivos pdf correspondientes a un diagrama de componentes con la visión general del sistema "Diagrama de componentes.pdfz los Diagramas de componentes administrador y nutricionista para ver en mejor detalle la conexión y composición interna de cada uno de los componentes de la arquitectura.
De igual forma se encuentran adjuntos los diagramas de secuencia de los componentes en el archivo "Diagrama de secuencias".

Con estos diagramas se especifica cómo está compuesto cada uno de los componentes definidos en la arquitectura seleccionada.

5.2. Base de Datos

Los atributos de calidad seleccionados para la base de datos son los siguientes:

Disponibilidad	Consistencia
Un fallo ocurre cuando el sistema no puede proporcionar el servicio que se espera de acuerdo con sus especificaciones afectando directamente a los usuarios. Las tácticas que se refieren a la disponibilidad del sistema se enfocan en tratar de conseguir que los errores existentes en un sistema no lleguen a convertirse en fallos o por lo menos limitar sus efectos y hacer posible la reparación [12].	Se refiere a la exigencia que cualquier transacciones de bases de datos Sólo debe cambiar datos afectados de formas permitidas. Los datos escritos en la base de datos deben ser válidos según todas las reglas definidas, incluyendo limitaciones, Cascades, factores desencadenantesy cualquier combinación de éstos. Esto no garantiza la exactitud de la operación de todas maneras hubiera querido programador de la aplicación (que es la responsabilidad del código de nivel de aplicación) pero simplemente que los errores de programación no pueden resultar en la violación de las reglas definidas.

Figura 3: Atributos de Calidad - Base de datos

Posteriormente, realizamos una comparación entre distintos tipos de modelos de bases de datos, como se muestra en la siguiente tabla:

	Ventajas	Desventajas
Modelo Relacional	• Cuenta con herramientas que garantizan la integridad referencial, lo que significa que, si se elimina un dato, todos los datos que dependían de este se eliminan también. Esto facilita la verificación de dependencias para el desarrollador. • Cuenta con herramientas que ayudan a la verificación de los registros duplicados. • Es un modelo de fácil comprensión y aplicación el cual funciona bastante bien para volúmenes de datos considerables.	• No es muy bueno para manejar archivos como multimedia, gráficos, etc. • En algunos casos se dificulta el manejo de cadenas como tipos de dato.
Orientado a Documentos	• El almacenar la información en archivos sin una estructura estricta hace más eficiente el manejo de grandes volúmenes de documentos. • La ausencia de relación facilita la recopilación de datos	• El modelado del código no recae en la base de datos, sino en la aplicación. • Limitaciones en las consultas. • Es difícil la combinación de una base de datos documental con otros modelos de bases de datos. • Se pierde la eficiencia cuando se trata de datos muy interconectados.
Orientado a Clave-Valor	• Almacenan los datos en diccionarios lo que facilita clasificarlos. • Por su forma de almacenar los datos, incrementa la velocidad de consulta a estos.	• No cuenta con estándares para el manejo de los datos de este modelo. • Presenta problemas al realizar trabajos de gran profundidad. • Las consultas se pueden hacer sólo desde la clave primaria, no se permite acceder ni organizar partiendo del valor.
Orientado a Familia de Columnas	• Cuando se realiza el procesamiento de una consulta, este modelo lee sólo los valores necesarios para realizar esta acción. Esto aumenta la eficiencia. • La información de cada columna se comprime, lo cual reduce el ancho de banda cuando se accede al disco.	• Las inserciones, actualizaciones o eliminación de datos requiere un mayor trabajo, pues se deben actualizar todos los pares clave-valor del registro que se consulta. Esto implica reducción de la eficiencia que se gana en la organización por columnas. • ebido a que debe consultar cada registro almacenado en la base de datos y cada tupla clave-valor y a pesar de que las consultas son rápidas, el proceso de generación de reportes no es muy eficiente

Figura 4: Comparación Modelos Base de datos

MODELO DE BD SELECCIONADO: Modelo Relacional.

Este modelo se escogió tendiendo en cuenta los atributos de calidad mencionados anteriormente y las ventajas que este modelo puede ofrecer sobre otros.

Este es un modelo cuya implementación no es compleja. A pesar de esto maneja buenos tiempos de respuesta a las consultas, lo que garantiza la consistencia de los datos al mantener actualizada la información en todas las pantallas del aplicativo. Por otra parte, es un modelo de base de datos bastante estable para la cantidad de datos y de peticiones que se van a tener en la aplicación, pues esta no es una aplicación que en el mercado pueda tener miles de peticiones por segundo, lo que garandiza que este modelo tenga una buena disponibilidad de mínimo 80 por ciento como se indicó en los requisitos.

Las entidades que se van a manejar son:
• Nutricionista: Indica los atributos que debe tener un nutricionista y gestiona las cuentas de los usuarios nutricionistas.
• Paciente: Indica los atributos que debe tener un paciente y gestiona los pacientes existentes pertenecientes a un nutricionista.
• Historia Clínica: Indica los campos base que debe tener una historia clínica y gestiona las historias de un paciente determinado.
• Preguntas Extra: Indica aquellas preguntas que un nutricionista agregó a determinada historia clínica de un paciente, junto con su respectiva respuesta en aquella historia.

5.2.1. Modelo Conceptual

El modelo conceptual de la base de datos será el siguiente:

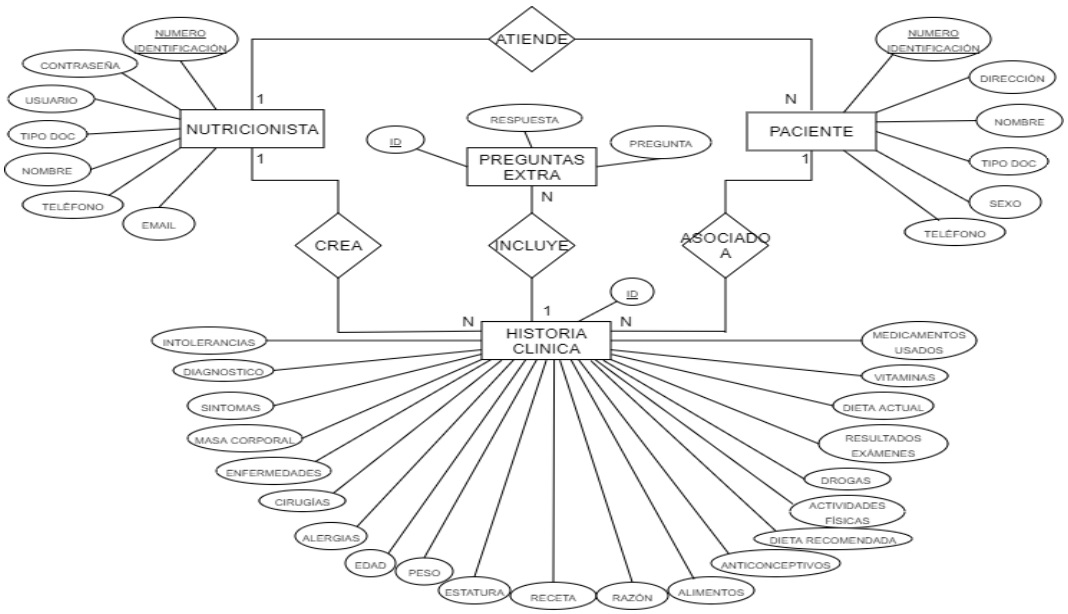


Figura 5: Modelo Conceptual

5.2.2. Modelo Lógico

El modelo lógico de base de datos será el siguiente:

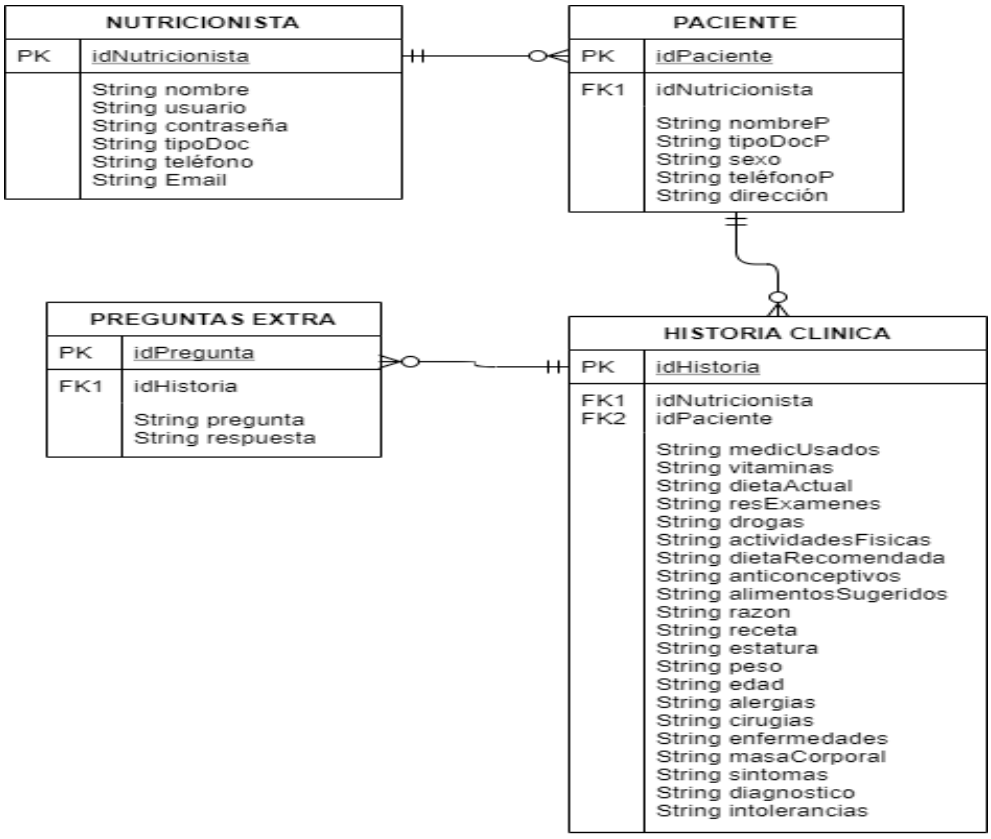


Figura 6: Modelo Lógico

6. Vista de Procesos

Para realizar la vista de procesos se escogieron dos procesos complejos:

1. Crear una historia clínica independiente.
2. Crear una nueva versión de una historia clínica.

Estos procesos se detallan en los diagramas adjuntos en la carpeta "Vista de procesos", cuyos nombres son "$Actividades - CrearHistoriaIndependiente.pdf$" y "$Actividades - NuevaversiondeHC.pdf$".

7. Vista Física

Diagrama - Arquitectura por capas:

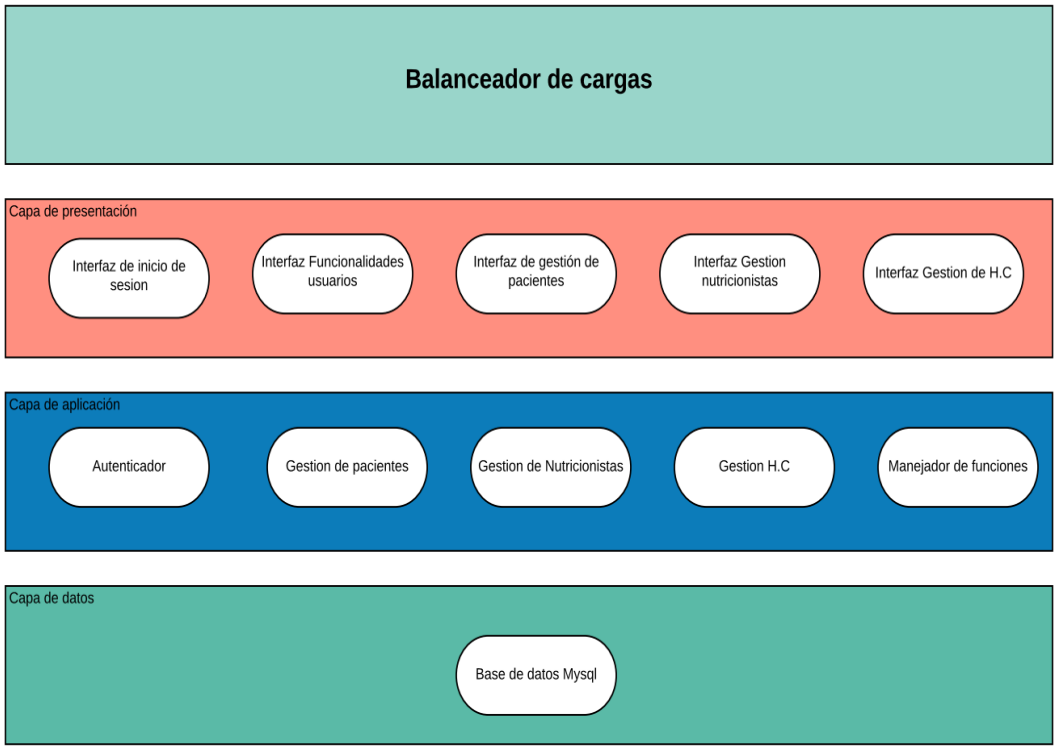


Figura 7: Arquitectura por capas

DOCUMENTACIÓN DE LOS COMPONENTES DE LA ARQUITECTURA:

CAPA DE PRESENTACIÓN:
Para la documentación de los componentes de esta capa, se adjuntan las pantallas iniciales del flujo correspondiente a cada componente:

Figura 8: Interfaz Inicio de Sesión

Figura 9: Interfaz Funcionalidades - Administrador

Figura 10: Interfaz Funcionalidades - Nutricionista

Figura 11: Interfaz de Gestión de Pacientes

Figura 12: Interfaz Gestión Nutricionistas

Figura 13: Interfaz Gestión de Historias Clínicas

CAPA DE APLICACIÓN:

Autenticador: Este componente será el encargado de verificar si el usuario que intenta ingresar hace parte del sistema, ya sea nutricionista o administrador.
Gestión de pacientes: Este componente será el encargado de atender las solicitudes del nutricionista cuando este necesite consultar a un paciente o agregarlo al sistema.
Gestión de nutricionistas: Este componente será el encargado de atender y realizar las solicitudes realizadas por el administrador cuando este realice consultas sobre los nutricionistas registrados en el sistema.
Gestión H.C: Este componente será el encargado de atender y ejecutar las solicitudes generadas por los nutricionistas cuando se realicen tareas relacionadas con el manejo de las historias clínicas de los pacientes registrados en el sistema.
Manejador de funciones: Este componente será el encargado de recibir las peticiones que se generen de parte del administrador y el nutricionista, para luego llamar al componente responsable de atender a dicha solicitud.

CAPA DE DATOS:

Base de datos Mysql: Este componente es donde se guardará la información del sistema por ejemplo (Credenciales, H.C, información pacientes,información nutricionista, etc).

8. Vista de Despliegue

El diagrama de despliegue es el siguiente:

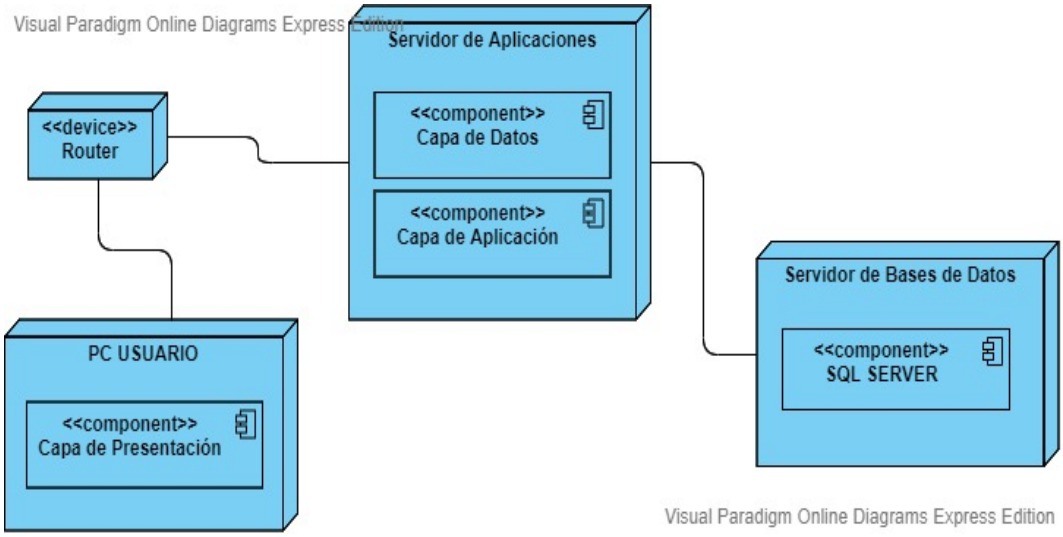


Figura 14: Diagrama de Despliegue del sistema

Se tendrá un computador que realice la gestión de los usuarios desde la interfaz gráfica del sistema.
Posteriormente, la información obtenida desde la interfaz local de un usuario debe atravesar un Gateway para poder conectarse con la nuve, lugar en el cual estará montado el Servidor de Aplicaciones.
Este último a su vez se conectará y consultará de un servidor el cual será el encargado de manejar todo lo de la base de datos del sistema.

DISPONIBILIDAD:
Tomando como promedio que un mes cuenta con 30 dias, esto equivale a un total de 720 horas al mes. Debido a que hay un máximo de 20 por ciento de no disponibilidad de la aplicacion, esto significa que a lo sumo 144 horas al mes no estaría disponible.

¿Cómo Garantizar que la aplicación estará disponible las otras 576 horas?
Existen dos componentes principales para garantizar la disponibilidad de la información en la aplicación web:

1. Asegurar que los sistemas operen de manera óptima.
2. Contar con copias de seguridad para prevenir la pérdida de información.

En cuanto al punto 2 sobre las copias de seguridad, esto no sólo garantiza la disponibilidad de la aplicación, sino que en caso de que la base de datos se caiga esta es una forma de recuperarla.

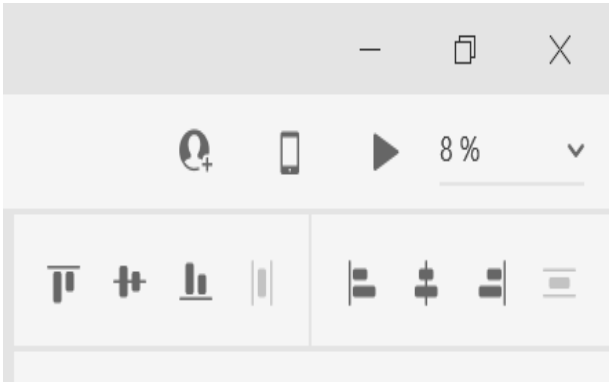
Para nuestro sistema en particular se harán copias de seguridad automáticas cada 24 horas, de forma que se pueda garantizar la disponibilidad y la recuperación de los datos.

9. Anexos

AVANCE DESARROLLO:
https://github.com/andresfelipedelgado/NustristoryApp

INTERFAZ Y FLUJO DEL SISTEMA:
Archivo ïnterfaz.xd"
Nota: El archivo debe ser abierto en Adobe XD
Posteriormente, se debe selecionar la primer pantalla que corresponde al login de la aplicación

y luego dar click en el botón de Play en la parte superior derecha



CARPETAS DEL PORTAFOLIO: (Los archivos enlistados son en formato PDF)

ARQUITECTURA POR CAPAS:
- Diagrama de clases
- Diagrama de componentes Administrador
- Diagrama de componentes Nutricionista
- Diagrama de componentes
- Diagrama de secuencias

CORRECCIONES 1ER ENTREGA:
- Documentación casos de uso
- Diagrama casos de uso - Admin
- Diagrama casos de uso - Nutricionista
- Documentación de Requisitos
- SRS

VISTA DE PROCESOS:

- Actividades - Crear Historia Independiente
- Actividades - Crear Nueva Version de HC