Architecture description template
for use with ISO/IEC/IEEE 42010:2011

Architecture Description of ‘Mejora al Sistema de Agendamiento’ for ‘RedSalud’

Version: 1.0

Document prepared by:
Lucas Bravo, Felipe Correa, Jairo González,
Carlos Mena, Lenny Roa, Francisco Vasquez

Contents

1 Introduction

Con el objetivo de abordar la problemática planteada por el equipo de Red Salud, hemos propuesto un software que busca mejorar significativamente la eficiencia operativa. Este informe proporciona una descripción detallada de la arquitectura del software, destacando sus componentes principales, la infraestructura tecnológica y las funcionalidades.

A través de una arquitectura escalable, este software está diseñado para adaptarse a las necesidades actuales y futuras de Red Salud, apoyando directamente a los enfermeros coordinadores en sus tareas.

1.1 Identifying information

La propuesta para este desarrollo de software está diseñada específicamente para Red Salud, con el objetivo de agilizar los sistemas de gestión por parte del enfermero coordinador de piso. Este proyecto está siendo elaborado siguiendo el modelo de vistas 4+1, metodología que nos permite abordad la arquitectura del sistema desde diferentes perspectivas. Además, estamos utilizando la aplicación de Figma para el prototipado, lo que nos permite crear interfaces de usuarios intuitivas y funcionales, mejorando la comunicación, colaboración entre el diseño y el desarrollo, asegurando las necesidades y expectativas de Red Salud.

1.2 Supplementary information

Junto a estudiantes de la Universidad San Sebastián, el tutor Rodrigo Caballero-Vivanco y el equipo de Red Salud, se ha estado trabajando para desarrollar una propuesta de software que resuelva y/o agilice las dificultades que tiene hoy en día el enfermero coordinador de boxs.

1.3 Other information

Para el desarrollo de esta propuesta se ha utilizado:

- El Modelo de vistas 4+1 de Kruchten

Mientras que para el modelo de trabajo utilizamos:

- Modelo de Prototipado

1.3.1 Overview (optional)

N/A

1.3.2 Architecture evaluations

La evaluación de la propuesta del software diseñado para Red Salud ha sido clasificada respecto a las entregas de progreso y entrega final de manera sumamente favorable, debido a la formalidad de nuestra propuesta.

1.3.3 Rationale for key decisions

-> Selección del Modelo de Vistas 4+1

- Con el fin de proporcionar una comprensión completa y detallada del sistema desde diferentes perspectivas, decidimos utilizar el modelo de vistas 4+1

-Como otra alternativa, también podríamos haber utilizado el Modelo C4

-Finalmente tomamos la decisión de utilizar el Modelo de Vistas 4+1, debido a que nos permite abordar la arquitectura del sistema mediante diferentes vistas, lo que facilita la comunicación y la compresión.

-> Figma para el Prototipado

-Con el fin de conseguir una herramienta eficiente y colaborativa para diseñar y validar las interfaces, utilizamos figma.

-Como otras alternativas, podríamos haber utilizado Adobe XD

- Figma fue seleccionado debido a su potente conjunto de características que facilitan el trabajo colaborativo en tiempo real, permitiendo diseñar y prototipar de manera eficiente.

1.3.4 Roles Designados

- Responsable del prototipo de software semi-funcional (figma.com): Felipe Correa y Lenny Roa

- Responsable del prototipo de infraestructura 3D: No necesitamos debido al tipo de propuesta

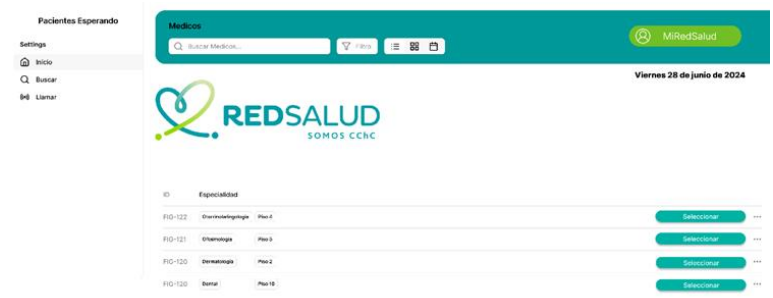
- Responsable de las heurísticas de usabilidad: Carlos Mena

- Responsable de los diagramas UML: Felipe Correa, Carlos Mena, Lucas Bravo
- Responsable del documento de diseño: Francisco Vasquez
- Responsable de presentar el hito final: Jairo Gonzales

1.4 Frames

A continuación, podremos observar cada uno de los frames creados para el prototipo de software diseñado para resolver la problemática planteada por Red Salud.

1.4.1 Frame Home

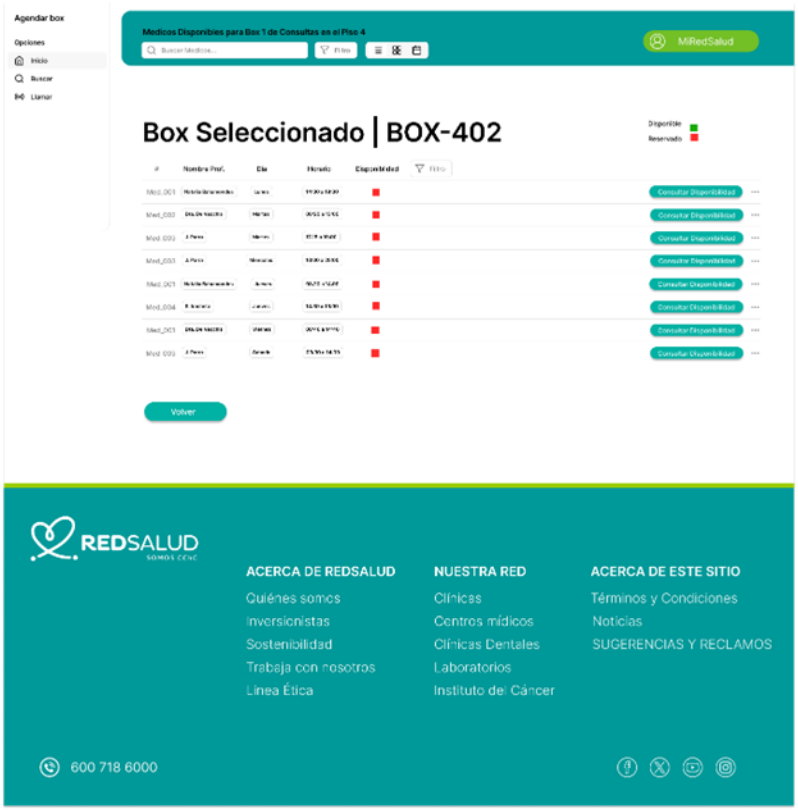


En nuestro primer frame, el coordinador de box podrá visualizar cada una de las especialidades y sus pisos correspondientes que se encuentren en el recinto de Red Salud.

1.4.2 Frame Box Disponibles por Especialidad

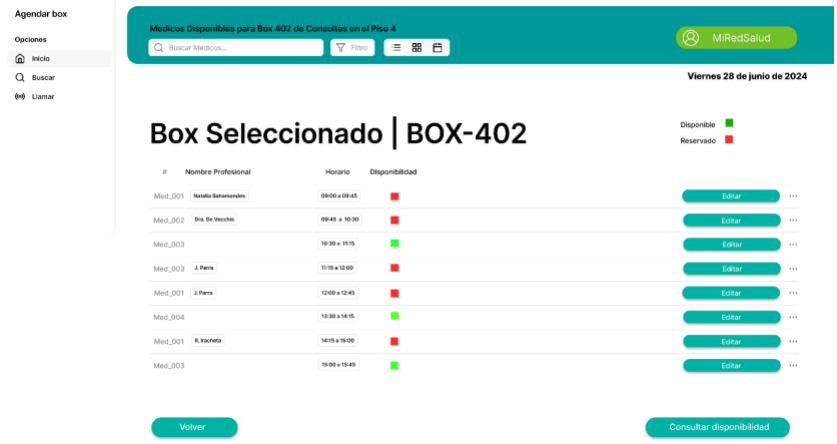
Luego de seleccionar una especialidad, el coordinador de box podrá visualizar la siguiente página, donde se podrá apreciar los boxes disponibles de la especialidad seleccionada, añadiendo además el piso en el cual se encuentra.

1.4.3 Frame Box Seleccionado



En este frame se puede apreciar el box seleccionado anteriormente en el frame 2, el propósito de esta pagina es poder ver la disponibilidad del box seleccionado y asimismo realizar una lista de cada nombre de profesional, su día y horario correspondiente en dicho box.

1.4.4 Frame Consultar Disponibilidad



En este último frame, se puede apreciar que realizará la misma función que en el frame 3, en el cual el coordinador de box podrá editar los horarios, disponibilidad y los profesionales ocupando dicho box realizando un flujo de consultas como también de reservas.

1.5 Heurísticas de Jacob Nilsen

En el contexto de la usabilidad y en interfaces de software, la heurística de Jacob Nilsen nos permite identificar necesidades que no son observables permitiéndonos así comprender de manera deductiva ciertos patrones o criterios que nos servirán para mejorar la experiencia del usuario.

1.5.1 Diagnosticar errores con retroalimentación y visibilidad del sistema

Se debe mantener informados a los usuarios sobre el diagnostico que se está haciendo mediante una retroalimentación adecuada y en un tiempo razonable.
Por ejemplo: “Cuando se está haciendo una acción sobre su celular y el usuario tiene que estar viendo que este todo bien”.

1.5.2 Coincidencia del mundo real y sistemas

Es cuando cambiamos asociaciones que vinculan el grupo de trabajo con el mundo real. El sistema debe hablar el lenguaje del usuario, usando palabras, frases y conceptos familiares en lugar de términos orientados al sistema.
Ejemplo: “Iconografías”.

1.5.3 Control y respuesta de usuario

Los usuarios a menudo eligen funciones del sistema por error y necesitan una “salida de emergencia” claramente marcada para salir del estado no deseado sin tener que pasar por un proceso prolongado.
Ejemplo: Funcionalidades como “deshacer” o “volver a la página anterior”.

1.5.4 Consistencia y Estándares

Los usuarios no deberían tener que preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Sigue convenciones de la plataforma. Se puede vincular con lo que es la responsividad, cuando una aplicación es responsiva es porque uno puede generar un uso que es muy similar, independiente lo que este usando.
Ejemplo: Botones de navegación ubicados de manera consistente y utilizando la misma terminología en toda la aplicación.

1.5.5 Prevención de Errores

Mejor que proporcionar buenos mensajes de error es un diseño cuidadoso que prevenga que ocurran errores en primer lugar. Advertir al usuario que está perdiendo la información y que no la está guardando.
Ejemplo: Confirmaciones antes de ejecutar una acción destructiva, como eliminar un archivo.

1.5.6 Reconocimiento en lugar de Recordación

Minimiza la carga de memoria del usuario haciendo visibles los objetos, acciones y opciones. El usuario no debería tener que recordar información de una parte del dialogo que a otra.
Ejemplo: Menús desplegables con opciones visibles en lugar de requerir que el usuario recuerde comandos específicos.

1.5.7 Flexibilidad y Eficiencia de Uso

Los aceleradores pueden a menudo acelerar la interacción para el usuario experto de tal manera que el sistema puede satisfacer tanto a usuarios inexpertos como avanzados. Es acomodarse a distintos estilos de trabajo, ya sea un usuario experto o novato.
Ejemplo: Atajos de teclado para usuarios avanzados.

1.5.8 Estética y Diseño Minimalista

Las interfaces no deben contener información irrelevante o que rara vez se necesita. Cada unidad extra de información en una pantalla compite con las unidades relevantes de información y disminuye su visibilidad relativa. Que tenga lo justo y necesario, además sea intuitivo para un uso necesario.
Ejemplo: Diseños limpios y simples sin elementos innecesarios

1.5.9 Ayuda a los Usuarios a Reconocer, Diagnosticar y Recuperarse de Errores

Los mensajes de error deben expresarse en un lenguaje sencillo, indicar precisamente el problema y sugerir una solución constructiva. Siempre se transparente, de la mejor manera posible al usuario que es lo que no está funcionando.

1.5.10 Ayuda y Documentación

Aunque es mejor que el sistema se pueda usar sin documentación, puede ser necesario proporcionar ayuda y documentación. Esta información debe ser fácil de buscar, centrada en la tarea del usuario, listar pasos concretos a realizar y no ser demasiado extensa.

2 Stakeholders and concerns

2.1 Stakeholders

En base a la información proporcionada por parte del equipo de Red Salud, podemos identificar que nuestro stakeholder es el enfermero coordinador de box.

El equipo de prototipado también es considerado un stakeholder

2.2 Concerns

Las principales preocupaciones de nuestro stakeholder se centran en los siguientes aspectos:

-Eficiencia

-Facilidad de Uso

-Efectividad

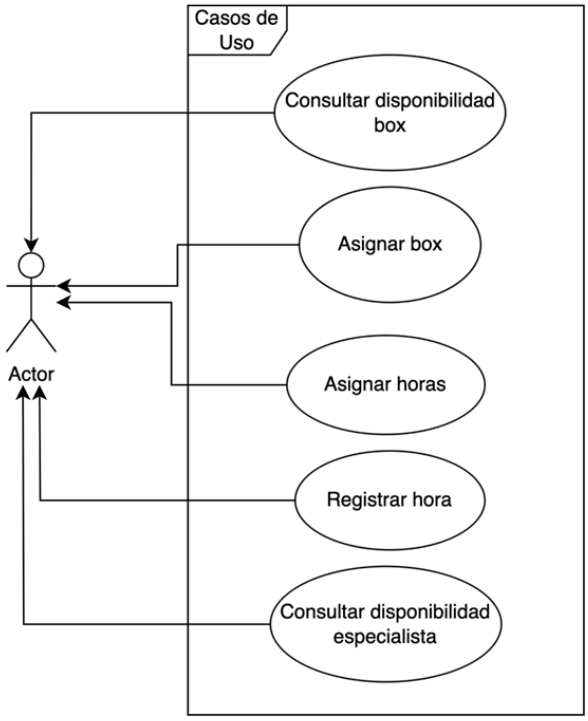
Es esencial que la propuesta de software optimice los procesos y recursos disponibles, garantizando una coordinación ágil y sin demoras. Asimismo, la interfaz debe ser intuitiva y fácil de usar, permitiendo que los enfermeros coordinadores de piso puedan operar el sistema sin dificultad. Finalmente, la efectividad del software es crucial para asegurar que todas las funcionalidades previstas se implementen correctamente.

2.3 Concern–Stakeholder Traceability

	enfermero coordinador de box
Eficiencia	X
Facilidad de Uso	X
Efectividad	X

3 Viewpoints+

3.1 Vista de Escenario



3.2 Overview

Representamos la vista de escenario mediante un diagrama UML de Casos de Uso. Esta vista permite identificar las funciones que ofrece el sistema y como los usuarios interactúan en él.

3.3 Concerns and stakeholders

Utilizamos el diagrama de casos de uso para capturar y representar de manera clara y visual como interactúa el enfermero coordinador de piso con el sistema. Este diagrama es fundamental para asegurar que el diseño del software aborde de manera efectiva las preocupaciones principales del stakeholder, como la eficiencia, facilidad de uso y la efectividad. Al detallar los casos de uso, podemos identificar y priorizar las funcionalidades esenciales que permitirán al enfermero coordinador realizar sus tareas de manera rápida y con menor esfuerzo, asegurando que la interfaz sea intuitiva y las operaciones del sistema sean efectivas y sin complicaciones.

3.3.1 Concerns

Los concerns que se pudieron identificar en las reuniones con el equipo de red salud son:

- Eficiencia
- Facilidad de Uso
- Efectividad

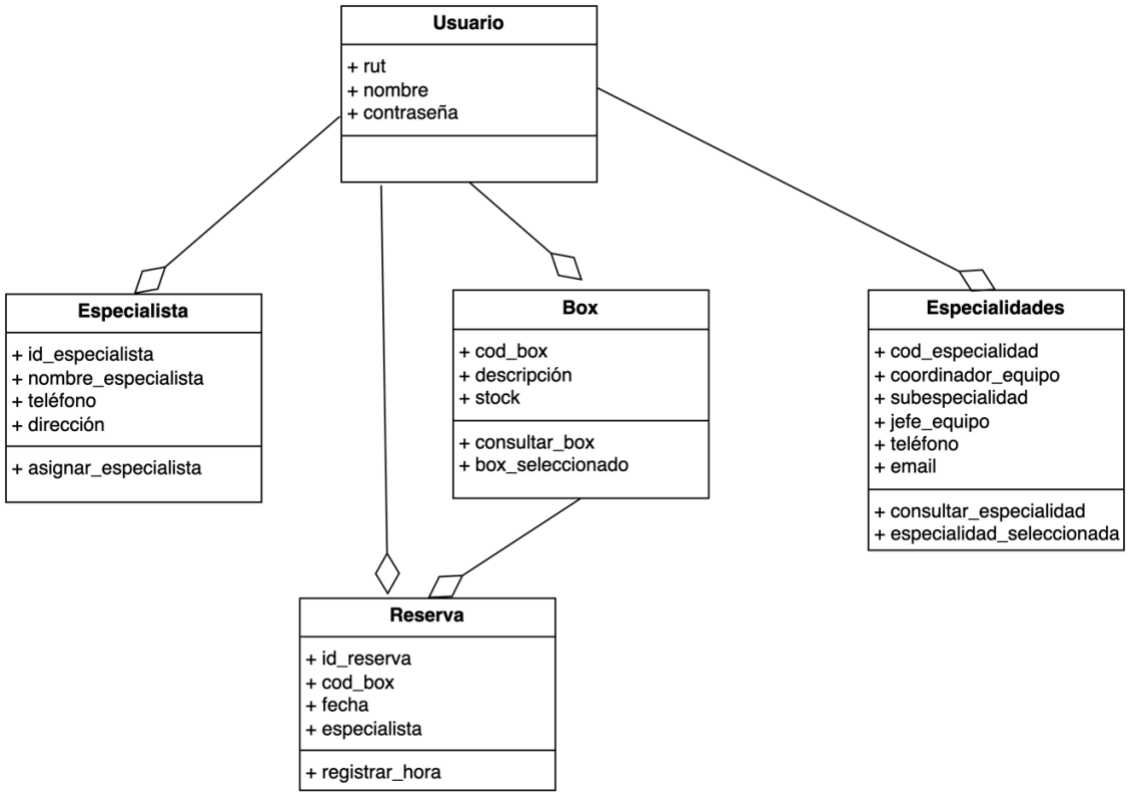
3.3.2 Typical stakeholders

- El enfermero coordinador de box va a ser nuestro typical stakeholder.

3.3.3 “Anti-concerns” (optional)

N/A

3.1.1 Vista Logica



3.2.1 Overview

Representamos la vista lógica mediante un diagrama UML de clases. Esta vista permite identificar la estructura del siste4ma y su operación, lo que es de utilidad para el usuario final

3.3.1 Concerns and stakeholders

Utilizamos el diagrama de clases para diseñar la estructura interna del sistema de manera organizada y coherente. Este diagrama nos permite definir claramente las relaciones entre los diferentes componentes del software, asegurando que todas las funcionalidades necesarias estén bien integradas. Al tener una estructura bien diseñada contribuye a la facilidad de uso del software, ya que facilita la implementación de una interfaz intuitiva y lógica. Además, la claridad y cohesión del diseño de clases aseguran la efectividad del sistema, permitiendo que todas las funciones operen sin problemas y soporten las actividades.

3.3.1.1 Concerns

Los concerns que se pudieron identificar en las reuniones con el equipo de red salud son:

- Eficiencia
- Facilidad de Uso
- Efectividad

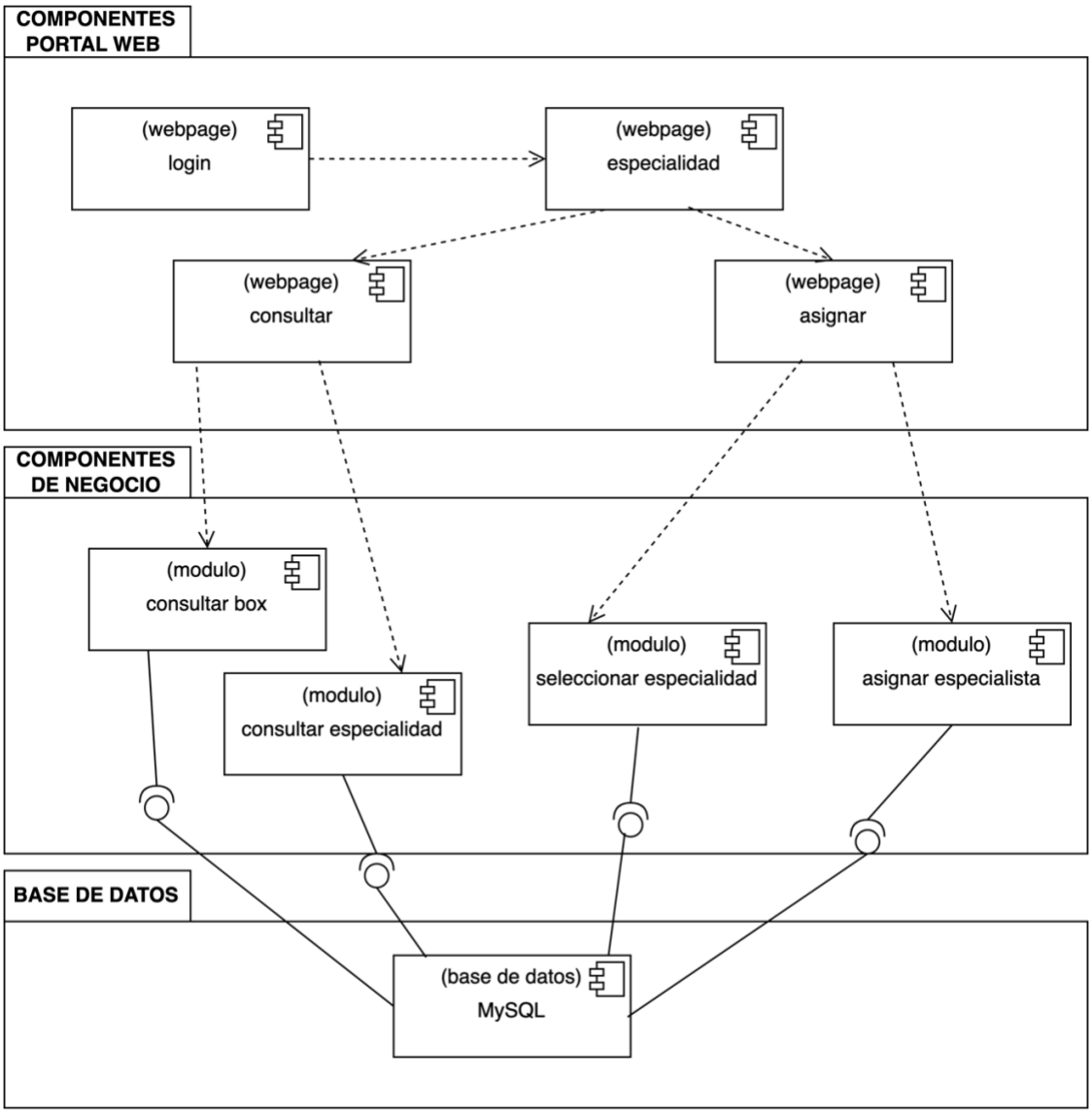
3.3.2.1 Typical stakeholders

- Para este punto de vista el stakeholder será el equipo de prototipado

3.3.3.1 "Anti-concerns" (optional)

N/A

3.1.2 Vista de Despliegue



3.2.2 Overview

Representamos la vista de despliegue mediante un diagrama UML de componentes. Esta vista permite identificar los componentes principales del sistema y como se relacionan entre si.

3.3.2 Concerns and stakeholders

Con el fin de mapear y organizar los diferentes módulos y subsistemas que constituyen el sistema, utilizamos un diagrama de componentes. Este diagrama es esencial para asegurar que cada parte del software funcione de manera eficiente y que las interacciones entre los componentes sean fluidas y sin conflictos. Esto significa un sistema más rápido y confiable, ya que la eficiencia se incrementa mediante las reducciones de redundancias y la optimización de las comunicaciones internas.

3.3.1.2 Concerns

Los concerns que se pudieron identificar en las reuniones con el equipo de red salud son:

- Eficiencia
- Facilidad de Uso
- Efectividad

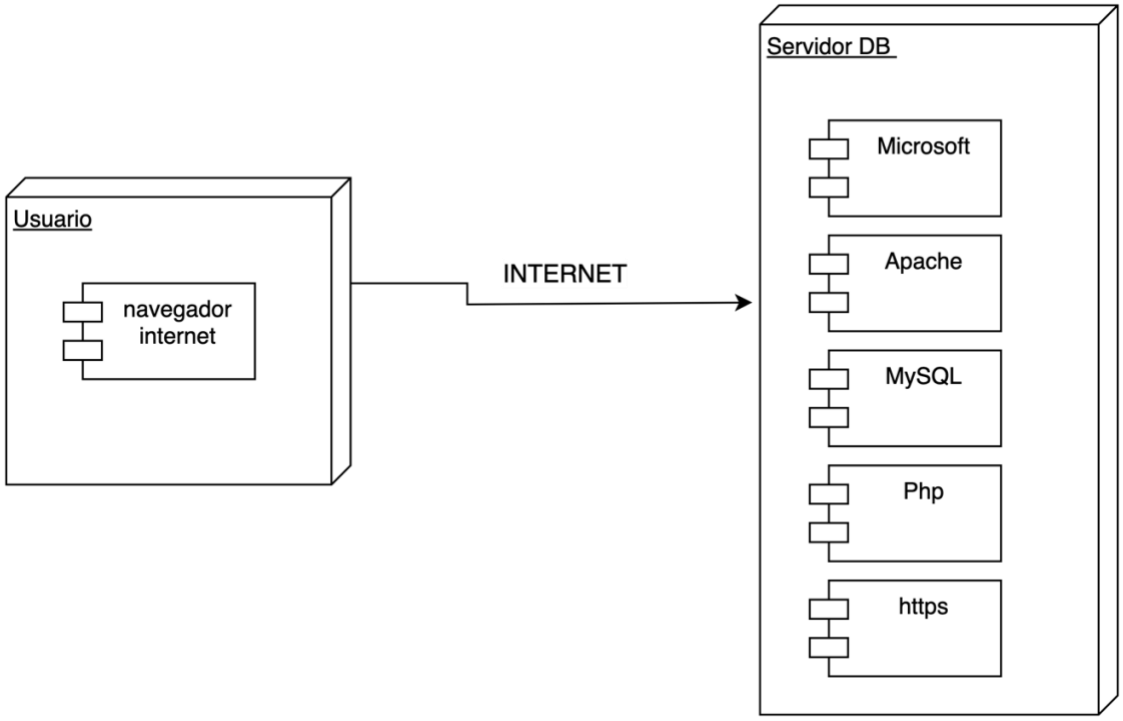
3.3.2.2 Typical stakeholders

- Para este punto de vista el stakeholder será el equipo de prototipado

3.3.3.2 “Anti-concerns” (optional)

N/A

3.1.3 Vista Fisica



3.2.3 Overview

Representamos la vista física mediante un diagrama de despliegue. Esta vista permite identificar donde será toda la infraestructura física.

3.3.3 Concerns and stakeholders

Utilizamos el diagrama de despliegue para representar la infraestructura física en la que se ejecutara el sistema de gestión de agenda, detallando como los componentes del software se distribuyen en el hardware disponible. Este diagrama es crucial para garantizar que el sistema opere de manera eficiente, optimizando el rendimiento y la utilización de recursos.

3.3.1.3 Concerns

Los concerns que se pudieron identificar en las reuniones con el equipo de red salud son:

- Eficiencia
- Facilidad de Uso
- Efectividad

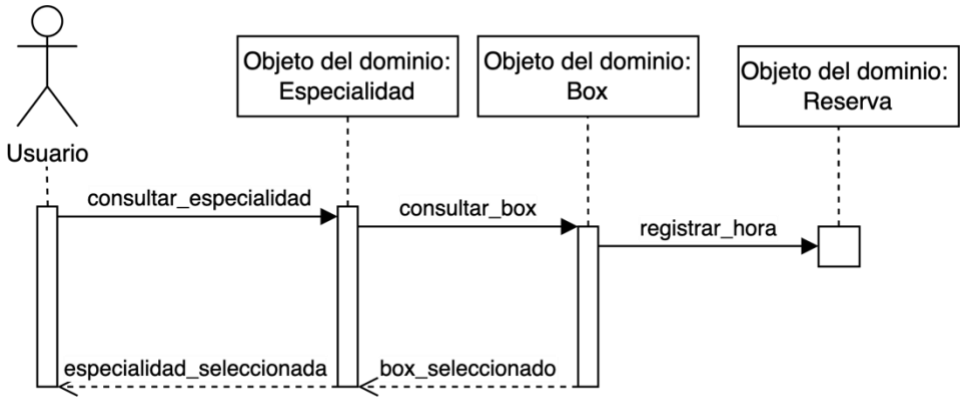
3.3.2.3 Typical stakeholders

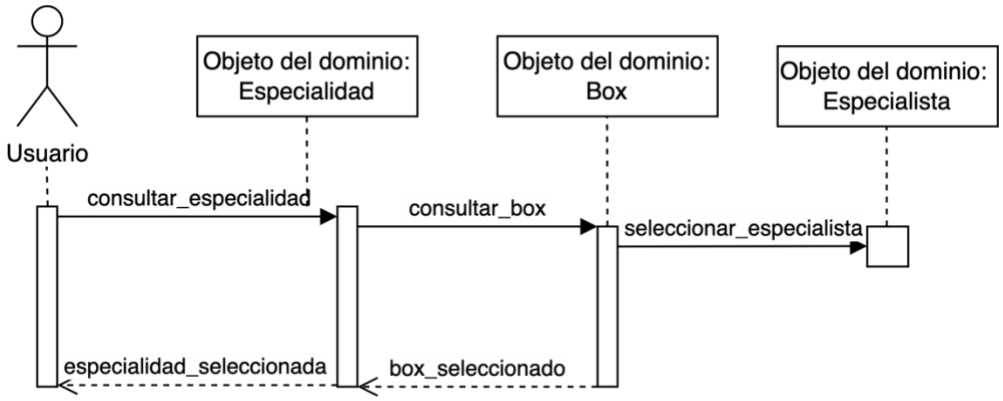
- Para este punto de vista el stakeholder será el equipo de prototipado

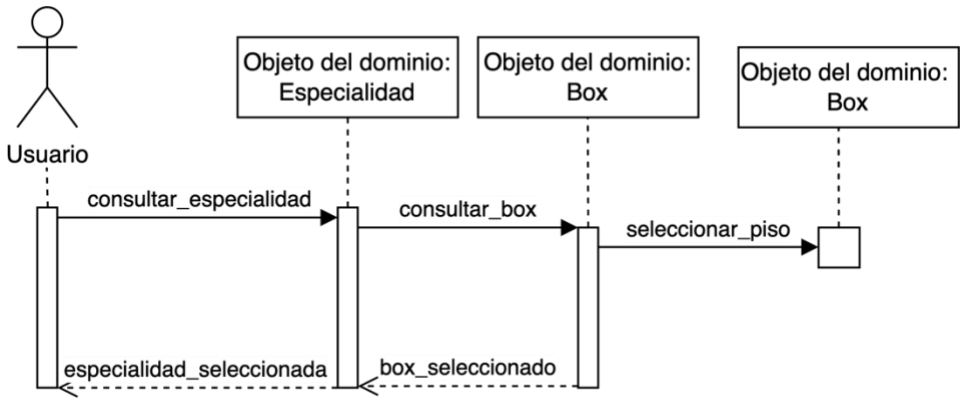
3.3.3.3 “Anti-concerns” (optional)

N/A

3.1.4 Vista de Procesos







3.2.4 Overview

Representamos la vista de procesos mediante un diagrama UML de secuencia. Esta vista permite identificar como se llevan a cabo las tareas y como fluye la información entre ellas.

3.3.4 Concerns and stakeholders

Para detallar como interactúan los diferentes componentes del sistema a lo largo del tiempo, utilizamos un diagrama de secuencia. Este diagrama es fundamental para asegurar que las operaciones y flujos de trabajo sean eficientes, minimizando los tiempos de espera y optimizando la coordinación entre procesos. Además, el diagrama de secuencia nos permite ayudar a identificar y solucionar posibles cuellos de botella o puntos de fallo en el proceso, asegurando que el sistema sea efectivo en la ejecución de todas sus funciones.

3.3.1.4 Concerns

Los concerns que se pudieron identificar en las reuniones con el equipo de red salud son:

- Eficiencia
- Facilidad de Uso
- Efectividad

3.3.2.4 Typical stakeholders

- Para este punto de vista el stakeholder será el equipo de prototipado

3.3.3.4 "Anti-concerns" (optional)

N/A

3.4 Model kinds+

Para el desarrollo de este prototipo de software se usaron 5 tipos de modelos los cuales son:

- Vista de Escenario: Diagrama de Casos de Uso
- Vista Lógica: Diagrama de Clase
- Vista de Despliegue: Diagrama de Componentes
- Vista Física: Diagrama de Despliegue
- Vista de Procesos: Diagrama de Secuencia

3.5 Modelo de Vista 4+1

3.5.1 Vista de Escenario: Diagrama de Casos de Uso

Empleamos el diagrama de casos de uso para capturar y representar de manera clara y visual la interacción del enfermero coordinador con el sistema propuesto. Al especificar los casos de uso, podemos identificar y priorizar las funcionalidades esenciales que permitirán al enfermero coordinador realizar sus tareas de forma rápida y efectiva, asegurando una interfaz intuitiva y que las operaciones del sistema sean efectivas y sin complicaciones.

I) Model kind languages or notations (optional)

N/A

II) Model kind metamodel (optional)

N/A

III) Model kind templates (optional)

N/A

3.5.1.2<Model Kind Name> operations (optional)

N/A

3.5.1.3<Model Kind Name> correspondence rules

N/A

3.5.2 Vista Lógica: Diagrama de Clases

Con el fin de tener una estructura del sistema organizada y coherente utilizamos un diagrama de clases. Este diagrama nos ayuda a definir claramente las relaciones entre los distintos componentes del software, garantizando que todas las funcionalidades esenciales estén integradas adecuadamente. Además, la claridad y cohesión del diseño de clases aseguran la efectividad del sistema, permitiendo que todas las funciones operen sin problemas y respalden las actividades.

I) Model kind languages or notations (optional)

N/A

II) Model kind metamodel (optional)

N/A

III) Model kind templates (optional)

N/A

3.5.2.2<Model Kind Name> operations (optional)

N/A

3.5.2.3<Model Kind Name> correspondence rules

N/A

3.5.3 Vista de Despliegue: Diagrama de Componentes

Con el fin de organizar los distintos módulos y subsistemas que conforman el sistema, empleamos un diagrama de componentes. Este diagrama es crucial para garantizar que cada parte del software opere eficientemente y que las interacciones entre los componentes sean de manera fluida y sin errores.

I) Model kind languages or notations (optional)

N/A

II) Model kind metamodel (optional)

N/A

III) Model kind templates (optional)

N/A

3.5.3.2<Model Kind Name> operations (optional)

N/A

3.5.3.3<Model Kind Name> correspondence rules

N/A

3.5.4 Vista Física: Diagrama de Despliegue

Se emplea un diagrama de despliegue para ilustrar la infraestructura física en la que se ejecutara el sistema propuesto para Red Salud, detallando como se distribuyen los componentes del software en el hardware disponible.

I) Model kind languages or notations (optional)

N/A

II) Model kind metamodel (optional)

N/A

III) Model kind templates (optional)

N/A

3.5.4.2<Model Kind Name> operations (optional)

N/A

3.5.4.3<Model Kind Name> correspondence rules

N/A

3.5.5 Vista de Procesos: Diagrama de Secuencia

Para detallar como interactúan los diversos componentes del sistema a lo largo del tiempo, utilizamos un diagrama de secuencia. Este diagrama es esencial para garantizar que las operaciones y flujos de trabajo sean eficientes, minimizando los tiempos y optimizando la coordinación entre los procesos. Además, podemos identificar y resolver posibles puntos de fallo en los procesos, asegurando que el sistema sea efectivo en la ejecución de todas sus funciones.

I) Model kind languages or notations (optional)

N/A

II) Model kind metamodel (optional)

N/A

III) Model kind templates (optional)

N/A

3.5.3.5<Model Kind Name> operations (optional)

N/A

3.5.3.5<Model Kind Name> correspondence rules

N/A

3.6 Operations on views

Para crear cada vista del sistema propuesto para Red Salud, seguimos una serie de pasos estructurados que aseguran un diseño completo y coherente. Primero, para el diagrama de casos de uso, identificamos y documentamos al actor principal, en este caso al enfermero coordinador de box, y los casos de uso que describen sus interacciones con el sistema. Luego, modelamos estas interacciones en diagramas que destacan las funcionalidades requeridas y los flujos de trabajo. Mientras que, en la vista lógica, comenzamos diseñando la estructura del sistema mediante el diagrama de clases, donde definimos las clases, sus atributos, métodos y relaciones, asegurando una base sólida para el desarrollo del software. Para la vista de desarrollo, organizamos los componentes del software en el diagrama de componentes, mostrando como se ensamblan y como interactúan entre sí, lo que facilita la implementación e integración del sistema. En la vista de procesos, creamos el diagrama de secuencia para representar los flujos de mensajes entre los componentes a lo largo del tiempo y a identificar posibles errores. Finalmente, en la vista física diseñamos el diagrama de despliegue para mapear la infraestructura del hardware y como los componentes del software se distribuyen en esta infraestructura, asegurando una correcta implementación y el uso eficiente de los recursos. Estos pasos, aplicados en cada vista, nos permite abordar las preocupaciones planteadas por el equipo de Red Salud, haciendo énfasis en la eficiencia, facilidad de uso y efectividad del sistema.

3.7 Correspondence rules

Las únicas reglas que se tuvieron que seguir fue respetar los nombres de los métodos al momento de hacer los UML, los métodos:

- consultar_especialidad
- consultar_box
- registrar_hora
- especialidad_seleccionada
- box_seleccionado
- seleccionar_piso

Son los métodos que siempre se tuvieron que seguir para. Hacer correctamente los UML.

3.8 Examples (optional)

N/A

3.9 Notes (optional)

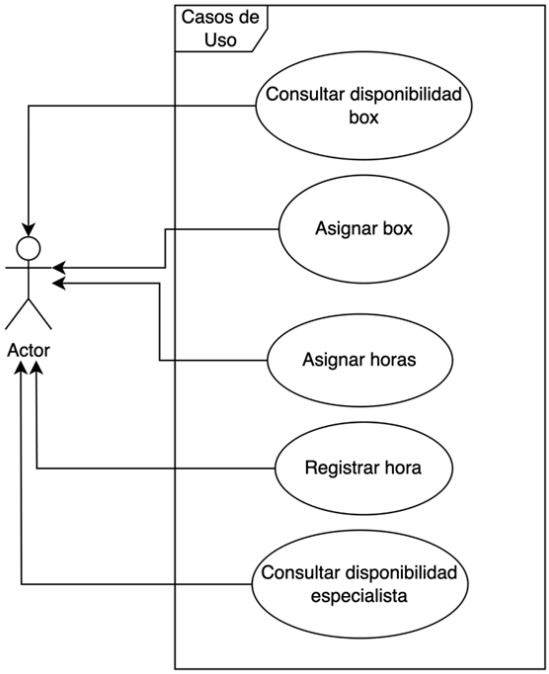
N/A

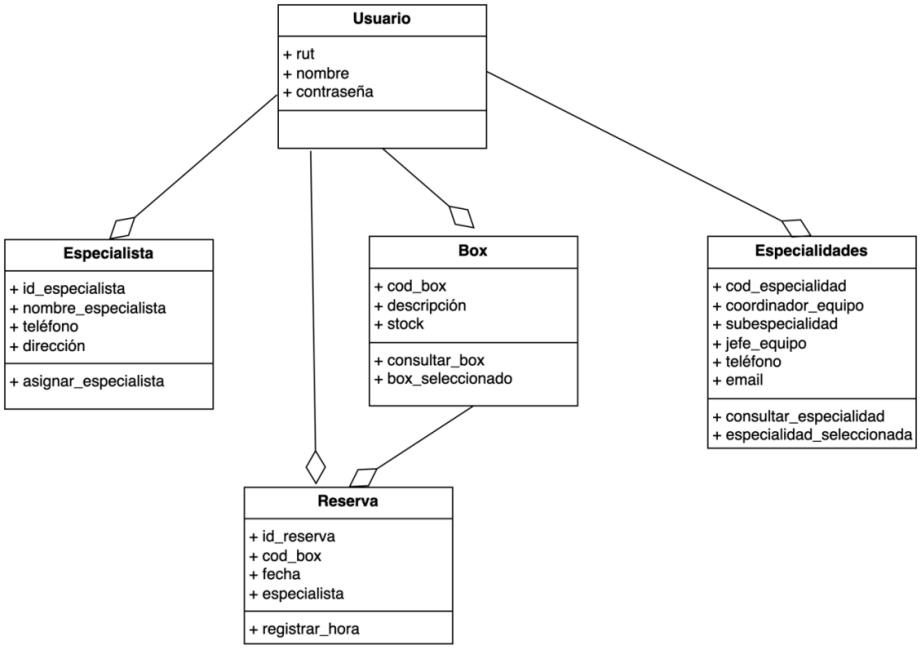
3.10 Sources

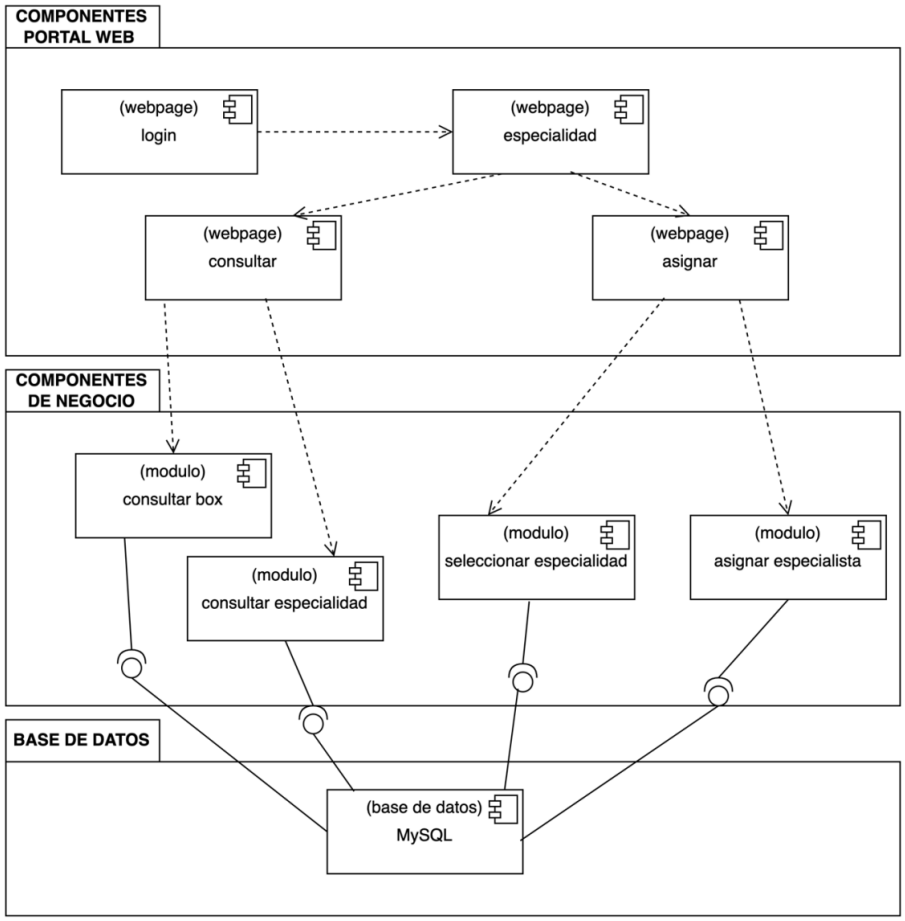
https://medium.com/javarevisited/4-1-architectural-view-model-in-software-ec407bf27258

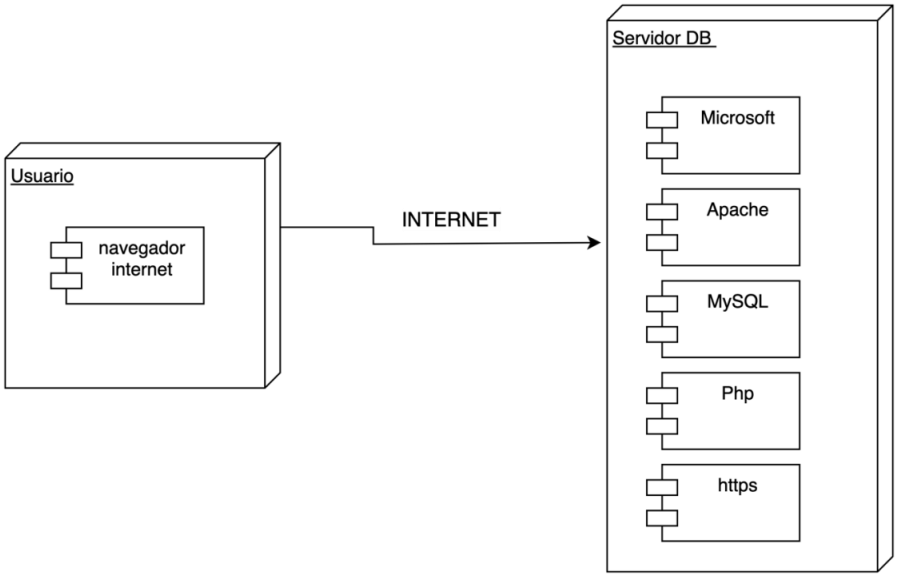
4 Views+

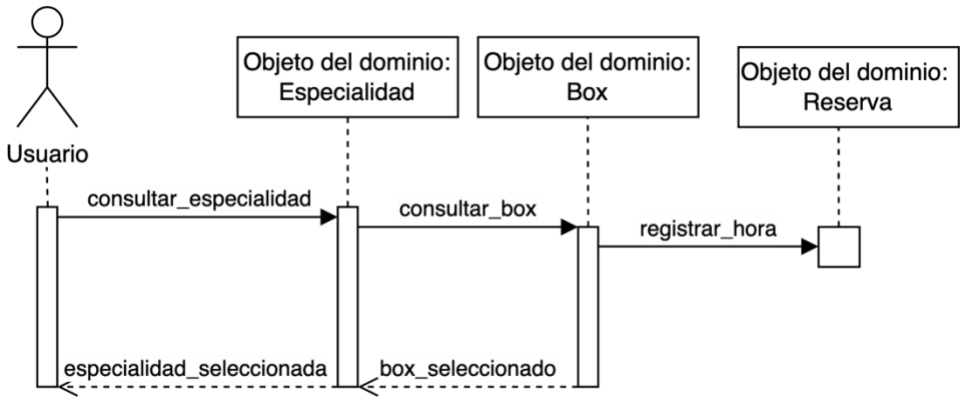
4.1 View: “The 4+1 View Model”

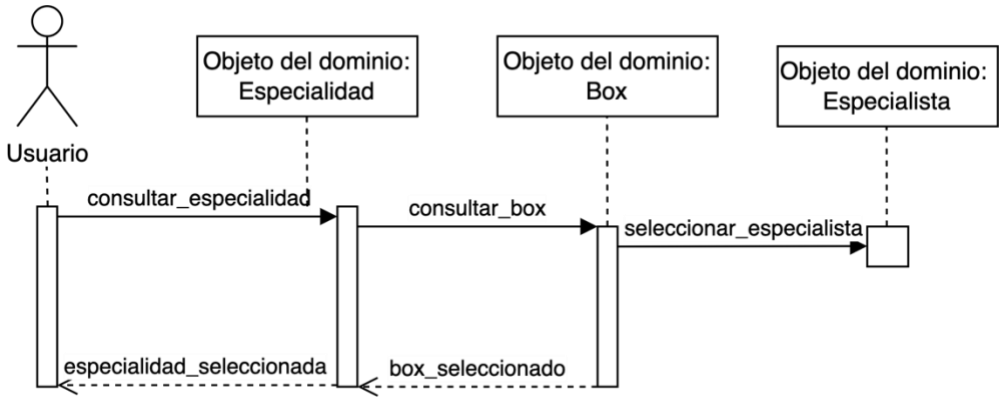


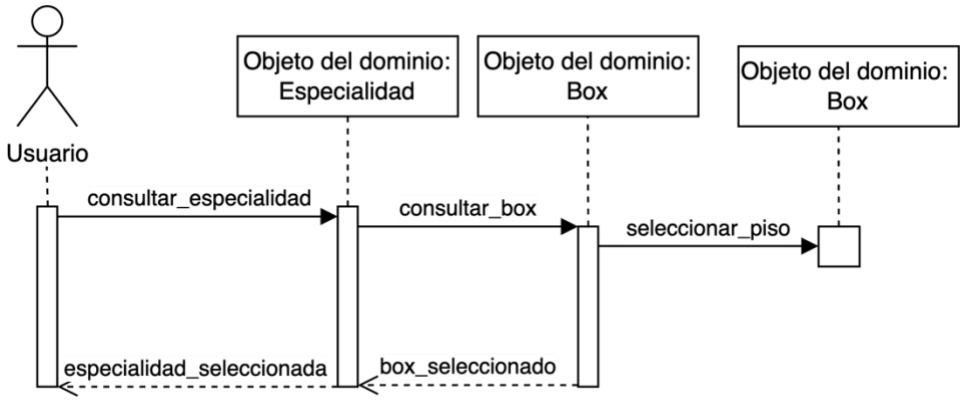



COMPONENTES PORTAL WEB
(webpage) login
(webpage) especialidad
(webpage) consultar
(webpage) asignar
COMPONENTES DE NEGOCIO
(modulo) consultar box
(modulo) consultar especialidad
(modulo) seleccionar especialidad
(modulo) asignar especialista
BASE DE DATOS
(base de datos) MySQL


Usuario
navegador internet
INTERNET
Servidor DB
Microsoft
Apache
MySQL
Php
https







4.1.1 Models+

El modelo que hemos utilizado para el desarrollo de esta propuesta es: “The 4+1 View Model” y usamos las siguientes vistas:

- Vista de Escenario: Donde se representa esta vista mediante un diagrama de Casos de Uso. Este permite identificar las funciones que ofrece el sistema y como los usuarios interactúan con él.

- Vista Lógica: Donde se representa esta vista mediante un diagrama de clases. Esta vista permite identificar la estructura del sistema y su operación, lo que es de utilidad para el usuario final.

- Vista de Despliegue: Para representar esta vista se utilizó un diagrama de componentes. Esta vista permite identificar los componentes principales del sistema y como se relacionan entre sí.

- Vista Física: Para representar esta vista se utilizó un diagrama de despliegue. Esta vista permite identificar donde será toda la infraestructura física.

- Vista de Proceso: Esta vista esta representada mediante un diagrama de secuencia. Esta vista permite identificar como se llevan a cabo las tareas y como fluye la información entre ellas.

4.1.2 The 4+1 View Model en el proyecto además del Modelo-Vista-Controlador

- Vista de Escenario: Esta vista fue seleccionada debido a que permite identificar las funciones adecuadas para el sistema y facilita como los usuarios interactúan con ella.

- Vista Lógica: Donde se representa esta vista mediante un diagrama de clases. Esta vista permite identificar la estructura del sistema y su operación, lo que es de utilidad para el usuario final.

- Vista de Despliegue: Para representar esta vista se utilizó un diagrama de componentes. Esta vista permite identificar los componentes principales del sistema y como se relacionan entre sí.
- Vista Física: Para representar esta vista se utilizó un diagrama de despliegue. Esta vista permite identificar donde será toda la infraestructura física.

- Vista de Proceso: Esta vista está representada mediante un diagrama de secuencia. Esta vista permite identificar como se llevan a cabo las tareas y como fluye la información entre ellas.

es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. este modelo es uno de los que utilizamos debido a la naturaleza de nuestro proyecto donde el modelo, el controlador y vista es desarrollada por nosotros para el organizador de boxes de Red Salud

4.1.3 Known Issues with View

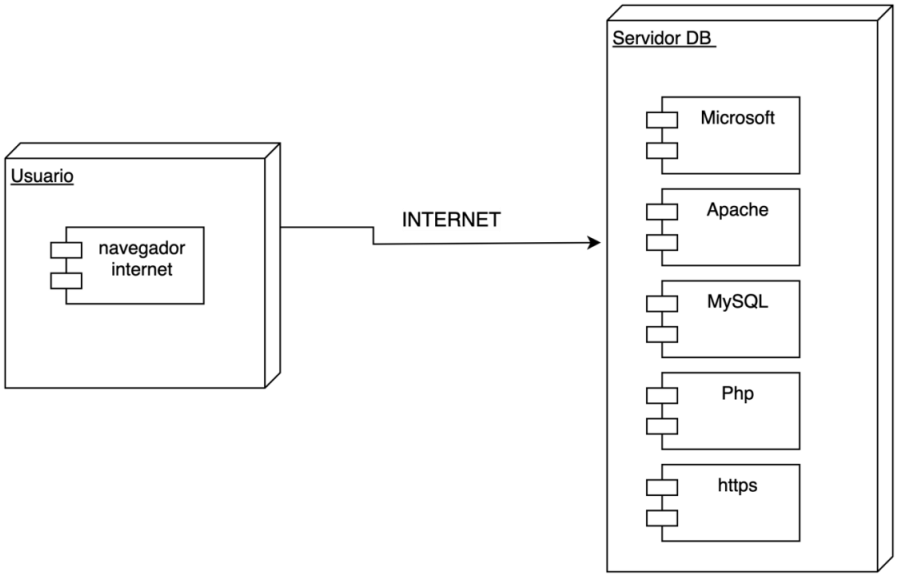
N/A

5 Consistency and correspondences

5.1 Known inconsistencies

N/A

5.2 Correspondences in the AD



La única relación directa esta mejor representada en el diagrama de despliegue porque de en claro la relación que hay entre el stakeholder y el navegador de internet con nuestra solución, además de la relación que hay entre nuestra página y el stack usado.

Las concerns en mente por parte de los stakeholders fueron la eficiencia, la facilidad de Uso y la efectividad, se unieron y se tuvieron en mente en todo momento al desarrollar las decisiones.

5.3 Correspondence rules

Las reglas que se tuvieron que seguir, como comentado en el punto 3.7 fue respetar los nombres de los métodos al momento de hacer los UML, los métodos:

- consultar_especialidad
- consultar_box
- registrar_hora
- especialidad_seleccionada
- box_seleccionado
- seleccionar_piso

Son los métodos que siempre se tuvieron que seguir para. Hacer correctamente los UML.

Autoevaluación y coevaluación

	L.B.	F.C	J.G	C.M	L.R.	F.V	N.Evaluado
L.B.	7.0	7.0	7.0	7.0	7.0	7.0	7.0
F.C.	7.0	7.0	7.0	7.0	7.0	7.0	7.0
J.G	7.0	7.0	7.0	7.0	7.0	7.0	7.0
C.M	7.0	7.0	7.0	7.0	7.0	7.0	7.0
L.R.	7.0	7.0	7.0	7.0	7.0	7.0	7.0
F.V	7.0	7.0	7.0	7.0	7.0	7.0	7.0
N.evaluador	7.0	7.0	7.0	7.0	7.0	7.0	7.0

Anexo

[1] Diseño de figma : https://www.figma.com/design/BSZdin7z7PYCie8m4pqtbL/Untitled?node-id=5-1142&t=MsTrSivsN4VLOUVq-0
[2] heurísticas de Nielsen : https://www.uifrommars.com/10-reglas-heuristicas-como-aplicarlas/
[3] Repositorio GitHub Contenido del informe: https://github.com/RobertoFichaz/RedSalud-Anexos

Bibliography

[1] Paul C. Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, and Judith Stafford. Documenting Software Architectures: views and beyond. Addison Wesley, 2nd edition, 2010.

[2] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering, 2(1):31–57, March 1992.

[3] IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, October 2000.

[4] ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description, December 2011.

[5] Alexander Ran. Ares conceptual framework for software architecture. In M. Jazayeri, A. Ran, and F. van der Linden, editors, Software Architecture for Product Families Principles and Practice, pages 1–29. Addison-Wesley, 2000.

[6] Nick Rozanski and Eo´in Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Addison Wesley, 2nd edition, 2011.

[7] Uwe van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. The Journal of Systems & Software, 85(4):795–820, April 2012.

[8] https://medium.com/javarevisited/4-1-architectural-view-model-in-software-ec407bf27258