Web de control de robots sanitarios

Florín Babusca, Pablo Molina, Juan Carlos Ruiz, Claudia Vega

14 de diciembre de 2022

Índice

1 Introducción

El producto a desarrollar consiste en una aplicación web que permita gestionar diferentes robots autónomos que pueda poseer un hospital, tales como robots de desinfección por luz ultravioleta, transporte de medicamentos, teleasistencia…

1.1. Objetivos

- Permitir la automatización de tareas triviales, pero costosas en tiempo
- Reducir el número de personal de centros hospitalarios
- Mejorar la calidad del servicio ofrecido a los pacientes
- Optimizar funciones del hospital para poder ser más seguras frente al SARS-CoV-2, tales como permitir la teleasitencia y el distanciamiento social.

1.2. Descripción de participantes y usuarios

- **Técnico**. Será el encargado de gestionar las capacidades y habilidades de los robots disponibles en el hospital.
- **Sanitarios**. Serán los usuarios que programarán actividades que desempeñarán los robots.

2

Descripción de requisitos

2.1. Justificación del formato

Hemos decidido utilizar dos formatos, uno tabular para los requisitos funcionales y otro para los requisitos no funcionales. Esto se ha hecho pensando en el modelo de trabajo que se utilizará durante el desarrollo de la web, ya que con total seguridad se aplicará la metodología SCRUM.

La principal razón de dicha decisión es porque durante el desarrollo web, es necesaria la interacción continua con los stakeholders, ya que en la mayoría de las veces dicho grupo presenta una heterogeneidad y mutidisciplinaridad en sus constituyentes muy amplia, la cual puede plantear retos para alcanzar consenso sobre los requisitos. Además, el entorno de desarrollo web es especialmente impredecible y dinámico, por lo que es necesario minimizar los riesgos asociados a dicha incertidumbre, y SCRUM lo consigue gracias a las entregas parciales que va generando y la constante participación de los stakeholders durante el desarrollo.

Al inicio de los requisitos (tanto funcionales como no funcionales) podremos observar una tabla con información relevante sobre estos, tales como, en caso de los requisitos funcionales, prioridad de cada requisito, que podrá ser Prioritario, Necesario o Deseable.

2.1.1. Formato de requisitos funcionales

Para los requisitos funcionales hemos decidido utilizar un modelo de tabla basado en User Stories, es así para que sea más comprensible para un cliente que no esté relacionado con el ámbito informático. En él se recogen la identificación del requisito (mediante el

código RF-n, siendo n el número del requisito), el usuario que utilizará dicha característica, el requisito en cuestión, el porqué del requisito, los criterios de aceptación que posee el requisito y la versión del mismo. En la tabla 2.1.1 podemos observar el modelo de tabla que se va a utilizar para los requisitos funcionales.

Cuadro 1: Modelo de requisito funcional

Atributo	**Descripción**
ID	RF-n
Cómo...	<Usuario>
Quiero...	<Funcionalidad>
Para obtener...	<Objetivo de la funcionalidad>
Criterio de Aceptación	<Criterio>
Versión	<número de versión>

El formato que se ha elegido para describir los requisitos funcionales es especialmente útil para SCRUM por la familiaridad que posee con los usuarios finales, ya que ayudará a la comprensión y validación de los mismos.

2.1.2. Formato de requisitos no funcionales

Los requisitos no funcionales se han representado también con un formato tabular, pero el contenido de la tabla asociado es diferente para facilitar el manejo dichos requisitos. En la tabla 2.1.2 podemos observar el formato que tendrán dichos requisitos.

Cuadro 2: Modelo de requisito no funcional

Atributo	**Descripción**
ID	RNF-n
Descripción	<Descripción>
Precedente	<ID Requisito Funcional>
Versión	<número de versión>

Dicho formato se ha elegido porque facilita mucho el seguimiento de la procedencia de dicho requisito no funcional, es decir, nos permite averiguar de qué requisito funcional es derivado y cómo se pueden relacionar con otros requisitos funcionales. Además, los requisitos no funcionales irán agrupados en categorías para así facilitar la lectura y el uso del documento.

2.2. Requisitos funcionales

A continuación, observamos en la tabla 2.2 la prioridad de los requisitos funcionales.

Cuadro 3: Requisitos funcionales

ID	Descripción	Prioridad
RF-1	Posibilidad de cancelar la ejecución de una tarea del robot	Necesario
RF-2	Poder asignar varias tareas a un robot	Deseable
RF-3	Asignar varios robots para una misma tarea	Deseable
RF-4	Los robots podrán notificar fallos durante la realización de una tarea	Prioritario
RF-5	Poder tener un filtro en el que seleccionar por tipos los robots que se muestran en la vista	Deseable
RF-6	Implementar tareas básicas para el robot y que sea escalable desde la interfaz.	Deseable
RF-7	Visualizar el estado de cada uno de los robots.	Necesario
RF-8	Guardar el historial de cada robot automáticamente.	Necesario
RF-9	En la monitorización del robot se deberá poder acceder a todos los tipos de tareas	Necesario
RF-10	Autorización mediante usuario-contraseña para entrar en la aplicación.	Deseable
RF-11	Los robots notifican a la aplicación si estos no pueden realizar la tarea por alguna razón.	Prioritario

Cuadro 4: Descripción requisito RF-1

Atributo	**Descripción**
ID	RF-1
Cómo...	Empleado sanitario
Quiero...	Poder cancelar en todo momento una tarea asignada a un robot determinado.
Para obtener...	Mayor control sobre la actividad del robot.
Criterio de Aceptación	Poder cancelar varias tareas sin que haya errores inesperados.
Versión	1.1

Cuadro 5: Descripción requisito RF-2

Atributo	**Descripción**
ID	RF-2
Cómo...	Empleado sanitario
Quiero...	Poder asignar varias tareas en secuencia para el robot.
Para obtener...	La posibilidad de asignar al robot una lista de tareas que la complete en un orden concreto.
Criterio de Aceptación	Poder asignar varias tareas sucesivas al robot y que este las ejecute en el orden que se dieron.
Versión	1

Cuadro 6: Descripción requisito RF-3

Atributo	**Descripción**
ID	F3
Cómo...	Empleado sanitario
Quiero...	Poder asignar a más de un robot para realizar la misma tarea.
Para obtener...	La capacidad de reducir el tiempo necesario de una tarea ya que la harán dos robots a la vez.
Criterio de Aceptación	Poder asignar a una misma tarea tres robots o menos.
Versión	1

Cuadro 7: Descripción requisito RF-4

Atributo	**Descripción**
ID	RF-4
Cómo...	Empleado sanitario
Quiero...	Poder ver si ha ocurrido un problema durante el desarrollo de la tarea del robot.
Para obtener...	Una información más detallada sobre cada trabajo que hayan llevado a cabo el robot.
Criterio de Aceptación	Poder visualizar el estado de fallo correctamente, siempre que se haya dado dicho problema.
Versión	1

Cuadro 8: Descripción requisito RF-5

Atributo	**Descripción**
ID	RF-5
Cómo...	Técnico Sanitario
Quiero...	Implementar un filtro en la aplicación.
Para obtener...	Seleccionar por tipos los robots que se muestran en la vista.
Criterio de Aceptación	Poder seleccionar y aplicar un único filtro (simultáneamente)
Versión	1.1

Cuadro 9: Descripción requisito RF-6

Atributo	**Descripción**
ID	RF-6
Cómo...	Técnico Sanitario
Quiero...	Implementar tareas básicas para el robot.
Para obtener...	Asignarles a los robots una o más tareas básicas.
Criterio de Aceptación	El objetivo es que así el sistema sea escalable a más tareas.
Versión	1

Cuadro 10: Descripción requisito RF-7

Atributo	**Descripción**
ID	RF-7
Cómo...	Empleado Sanitario
Quiero...	Visualizar el estado de cada uno de los robots.
Para obtener...	Información acerca de cuál está libre.
Criterio de Aceptación	Que sea intuitivo y fácil de utilizar para el usuario medio.
Versión	1

Cuadro 11: Descripción requisito RF-8

Atributo	**Descripción**
ID	RF-8
Cómo...	Técnico Sanitario
Quiero...	Guardar el historial de cada robot automáticamente.
Para obtener...	Información acerca de los eventos pasados del robot.
Criterio de Aceptación	Que se guarden los datos de forma diferenciada e inequívoca
Versión	1

Cuadro 12: Descripción requisito RF-9

Atributo	**Descripción**
ID	RF-9
Cómo...	Técnico sanitario
Quiero...	Poder ver y gestionar en la monitorización del robot los tipos de tareas que este pueda desempeñar.
Para obtener...	Una visión general de las diferentes tareas que puede desempeñar un tipo de robot y poder modificarlas en caso de que sea necesario
Criterio de Aceptación	Obtener una vista de todas las tareas posibles de un robot y poder manipularlas de manera sencilla
Versión	2

Cuadro 13: Descripción requisito RF-10

Atributo	**Descripción**
ID	RF-10
Cómo...	Técnico sanitario
Quiero...	Que los usuarios de la aplicación puedan entrar a la aplicación con un inicio de sesión básico (ID, contraseña y rol)
Para obtener...	Mayor seguridad en el acceso a la aplicación
Criterio de Aceptación	Que la aplicación no sea accesible por usuarios sin permisos
Versión	1

Cuadro 14: Descripción requisito RF-11

Atributo	**Descripción**
ID	RF-11
Cómo...	Empleado sanitario
Quiero...	Que los robots notifiquen a la aplicación que no pueden realizar la tarea por alguna razón
Para obtener...	Información necesaria y rápida sobre los robots en ese estado
Criterio de Aceptación	Que la aplicación reciba el aviso de que el robot no ha podido ejecutar la tarea enmendada
Versión	1

2.3. Requisitos no funcionales

Observamos en la tabla 2.3 la prioridad de los requisitos no funcionales, así como su tipo.

Cuadro 15: Requisitos no funcionales

ID	Descripción	Tipo	Prioridad
RNF1	Se deberá proporcionar un sistema con interfaz de usuario	Interfaz de usuario	Prioritario
RNF2	El programa debe tener una naturaleza sencilla para poder configurar o manejar a distintos tipos de robots	Usabilidad	Necesario
RNF3	Se debe poder ver lo que estan haciendo los robots de un vistazo	Usabilidad	Prioritario
RNF4	La aplicación debe soportar al menos dos usuarios simultáneamente	Mantenibilidad	Necesario
RNF5	Deberán existir distintos tipos de error para el robot	Fiabilidad	Deseable
RNF6	La información asociada a una tarea deberá incluir la fecha y la hora a la que se ha realizado	Usabilidad	Deseable
RNF7	Solo se almacenará el historial de la última jornada	Usabilidad	Deseable
RNF8	Recibir información de los estados de los robots con una asincronía	Interfaz de Usuario	Deseable

2.3.1. Requisitos de Interfaz de Usuario

Cuadro 16: Descripción requisito RNF-1

Atributo	Descripción
ID	RNF-1
Descripción	Se deberá proporcionar un sistema con interfaz de usuario
Precedente	RF-7
Versión	1

Cuadro 17: Descripción requisito RNF-8

Atributo	**Descripción**
ID	RNF-8
Descripción	Recibir información de los estados de los robots con una asincronía máxima de 30 segundos
Precedente	RF-7
Versión	1

2.3.2. Requisitos de Usabilidad

Cuadro 18: Descripción requisito RNF-2

Atributo	**Descripción**
ID	RNF-2
Descripción	El programa debe tener una naturaleza sencilla para poder configurar o manejar a distintos tipos de robots
Precedente	RF-2
Versión	1

Cuadro 19: Descripción requisito RNF-3

Atributo	**Descripción**
ID	RNF-3
Descripción	Se debe poder ver lo que están haciendo los robots de un vistazo
Precedente	X
Versión	1

Cuadro 20: Descripción requisito RNF-6

Atributo	**Descripción**
ID	RNF-6
Descripción	La información asociada a una tarea deberá incluir la fecha y la hora a la que se ha realizado.
Precedente	RF-8
Versión	1

Cuadro 21: Descripción requisito RNF-7

Atributo	**Descripción**
ID	RNF-7
Descripción	Solo se almacenará el historial de la última jornada.
Precedente	RF-8
Versión	1

2.3.3. Requisitos de Mantenibilidad

Cuadro 22: Descripción requisito RNF-4

Atributo	**Descripción**
ID	RNF-4
Descripción	La aplicación debe soportar al menos dos usuarios simultáneamente
Precedente	X
Versión	1

2.3.4. Requisitos de Fiabilidad

Cuadro 23: Descripción requisito RNF-5

Atributo	**Descripción**
ID	RNF-5
Descripción	Se deberá mostrar los mensajes de error que se hayan producido de forma amigable con el usuario
Precedente	RF-4
Versión	1.1

3 Modelado

Una vez definidos los requisitos del cliente y con la información expuesta en la sección 2 podemos comenzar a idear el funcionamiento de nuestra página web.

En la siguiente sección vamos a modelar el funcionamiento de nuestra web en diversas áreas. En primer lugar, presentamos varios modelos IFML (*Interaction Flow Modeling Language*)[1] en los que se representa el flujo de comportamiento de nuestra aplicación web. A continuación usaremos UML (*Unified Modeling Language*)[2] para mostrar las relaciones entre las diferentes entidades presentes en la aplicación y cómo se comunican entre ellas. Por último, vemos un diagrama de despliegue en el que vemos como se realiza el lanzamiento de la aplicación para poder ser usada por el usuario.

3.1. Modelo en IFML

Para realizar el modelo IFML de nuestra aplicación, que pretende definir el comportamiento de la misma en función de las acciones del usuario, nos hemos basado en los requisitos descritos en la sección 2. Esto es así para intentar cumplir al máximo de nuestras posibilidades las expectativas del cliente.

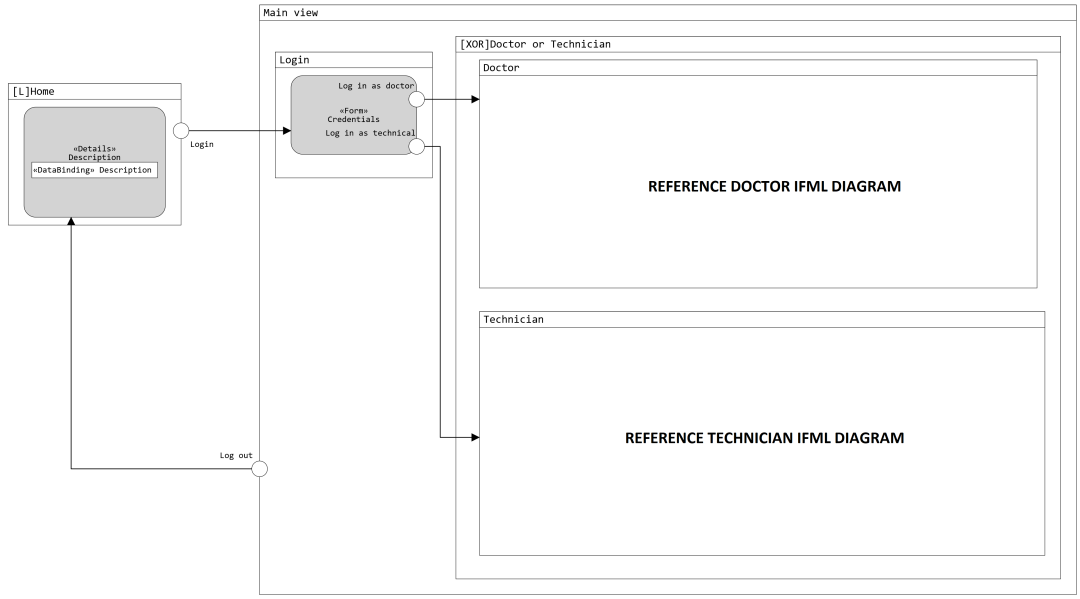

Figura 1: Modelo IFML general de la aplicación. Véase las figuras 2 y 3 para completar la información

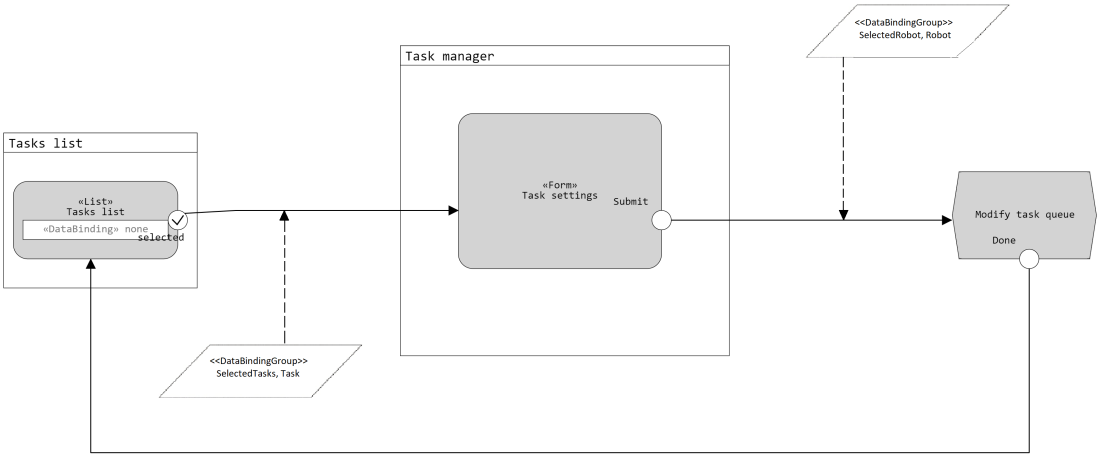

Figura 2: Modelo IFML concreto de la vista del médico

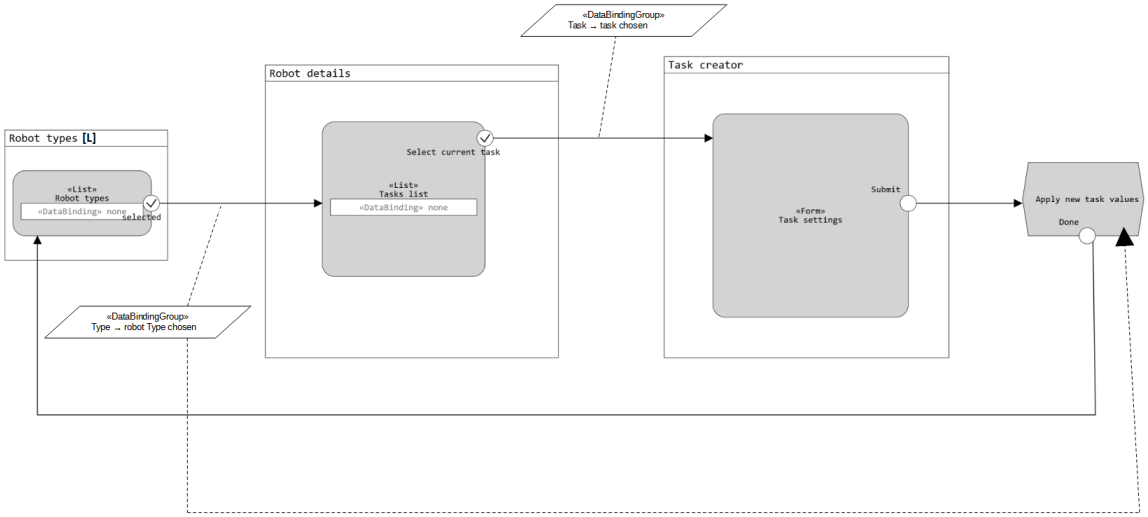

Figura 3: Modelo IFML concreto de la vista del técnico

Como podemos observar en la figura 1, la web dispondrá de una pantalla de inicio, con la posibilidad de iniciar sesión, lo que representa el requisito F-10.

Dentro del contenedor *Main View*, tendremos diferentes *View Containers* dependiendo si el usuario es un empleado sanitario o bien, un técnico.

Si el usuario es un personal sanitario, usará el contenedor *Doctor*. En él, el usuario podrá visualizar una lista con todos los robots y sus respectivos estados (ocupado, libre o error). Dicha característica implementa los requisitos funcionales RF-4, RF-7 y RF-12.

Además, el usuario podrá seleccionar un robot en concreto de dicha lista, lo cual desplegará la información completa de ese robot (su tarea actual, problemas surgidos y el historial de tareas de la última jornada), y también permitirá asignar una nueva tarea o bien cancelarla. Si el usuario selecciona un elemento del historial (elemento de *History of previous tasks*), se mostrará la configuración de la tarea, y si dicha tarea está en ejecución, se podrá cambiar la configuración de la misma. Las características mencionadas reflejan los requisitos funcionales RF-1, RF-2, RF-3, RF-4, RF-7, RF-8, RF-9 Y RF-11.

Todas las acciones de gestión de tareas descritas anteriormente se actualizarán en el robot afectado con la acción *Modify task queue* y se volverá a la lista de robots anterior.

Si el usuario es un técnico, obtendremos un comportamiento similar al anterior. En este caso, el contenedor a usar será *Technician*. En él, el usuario podrá visualizar una lista con todos los tipos de robots existentes. Dicha característica implementa el requisito RF-5.

Si el técnico elige un determinado tipo, se mostrará un formulario con todas las posibles tareas que dicho tipo de robot puede ejecutar. Además, permitirá modificar, eliminar y añadir tareas. Para ello, se desplegará un formulario de casillas con dos columnas y diez filas donde el técnico podrá añadir hasta 10 pares clave-valor. Es decir, si el robot dispone del atributo DurationOfTask y tiene como valor el tiempo en segundo de la ejecución de una tarea, el técnico añadirá en la casilla de la primera columna DurationOfTask y en la casilla de la segunda columna de la misma fila, añadirá el tiempo en segundos.

Dichas características representan los requisitos funcionales RF-6 y RF-9.

Todas las acciones de gestión descritas anteriormente se actualizarán en los robots del mismo tipo con la acción *Apply new task values* y se volverá a la lista de tipos de tarea anterior.

3.2. Diseño de lógica

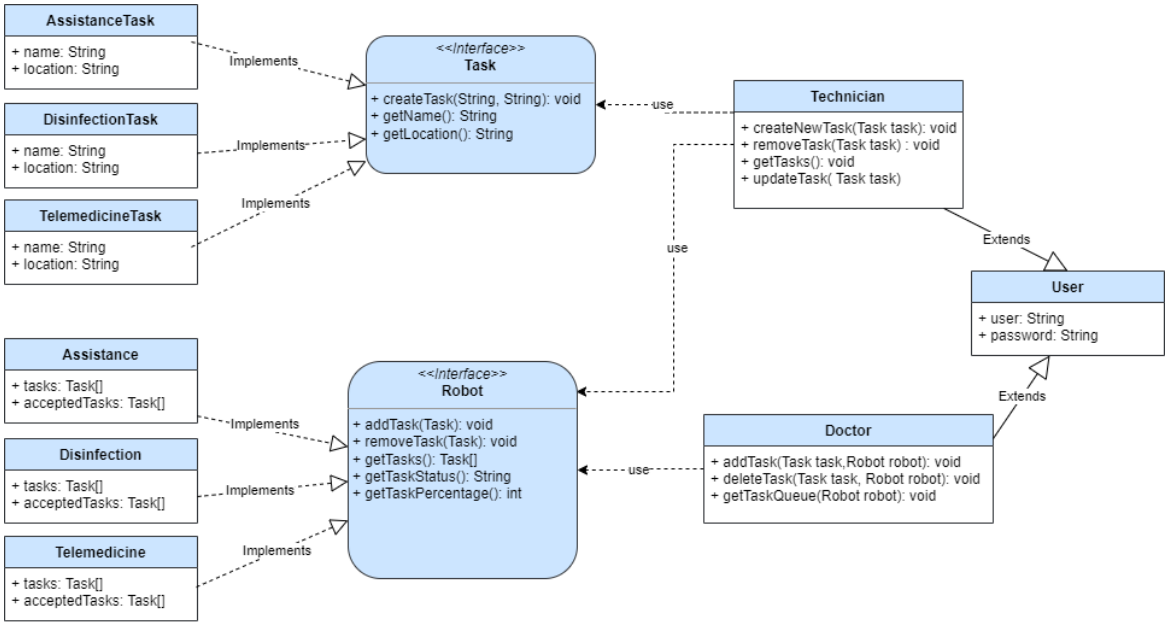

Figura 4: UML class diagram

Tal y como podemos ver en la Figura 4, el sistema estará formado por la clase *User*, la cual guardará el usuario y contraseña. Esta clase tiene dos hijos: *Technician* y *Doctor*.

La clase *Technician* se relaciona con la interfaz de tareas *Task*, mediante relaciones de uso para crear, eliminar o modificar el funcionamiento de las tareas, y con la interfaz *Robot*,

para poder crear nuevos tipos de robots. *Doctor* también tiene una relación de uso con la interfaz *Robot* para poder añadir o eliminar tareas de la cola de tareas correspondiente al robot que se quiera modificar. La interfaz *Robot* será implementada, en principio, por tres tipos de robots: *Assistance*, *Disifection* y *Telemedicine*. Sin embargo, el modelo permite crear facilmente nuevos tipos de robots con diferentes funcionalidades y nos ofrece una mayor escalabilidad. Esto es así también para las tareas, ya que la interfaz *Task* nos permite crear gran cantidad de tareas interoperables sin problema. Como podemos observar, se cumplen casi por completo los requisitos funcionales requeridos por el usuario, lo que hace indicarnos que con este diagrama estamos eligiendo el camino correcto en nuestro desarrollo del proyecto.

Las acciones mostradas en el IFML se realizarán de la siguiente forma desde las clases:

- **Modify task queue**. Para ello, se usará la clase Doctor en la que se pasarán como parámetros el robot seleccionado y se realizarán las tareas seleccionadas por el usuario, es decir, si el empleado sanitario añade una tarea a un robot, se usará el método *addTask*, para eliminar una tarea asignada se usará el método *deleteTask* y para consultar la información de un robot, se usará el método *getTaskQueue*.

- **Apply new tasks values**. Para ello, se usará la clase Technician. Si el técnico desea crear un nuevo tipo de tarea, se usará el método *createNewTask*, para eliminar tipo de tareas, se usará el método *removeTask* y para poder visualizar todos los tipos de tarea para un robot en concreto, se usará el método *getTasks*. Finalmente, si el técnico desea actualizar el funcionamiento de alguna tarea en concreto de un robot, usará el método *updateTask*.

3.3. Diagrama de despliegue

Por último, en la figura 5 podemos observar el diagrama de despliegue que tendrá nuestra web. Al ser de uso local en los hospitales en los que vaya a utilizarse, no necesitaremos del uso de bases de datos, podremos guardar toda la información necesaria en el disco local de la máquina servidor. Esto ayudará a la web a ser mucho más rápida y eficiente, ya que una web estática necesita de menos recursos para su funcionamiento y además no sufre de bajas latencias cuando se produzcan caídas de conexión. Por otro lado, podemos observar

que se utilizará HTML para utilizar la web, lenguaje que todos los navegadores conocen y pueden mostrarlo al usuario. Por último, el servidor web alojará una aplicación de flask que será la encargada de realizar todos los procesos necesarios para el funcionamiento de la web.

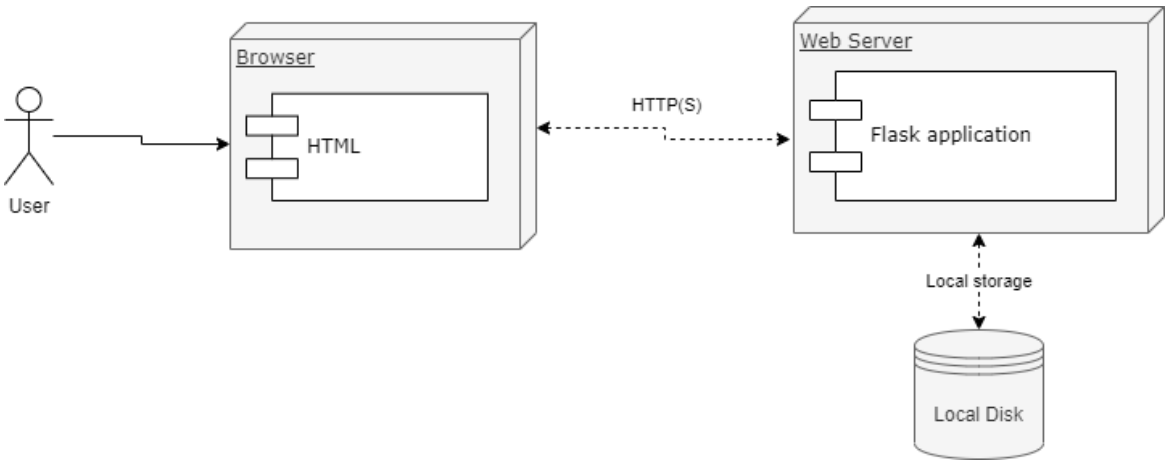

Figura 5: Diagrama de despliegue

4
Usabilidad

En esta sección vamos a realizar un informe sobre la usabilidad de nuestra aplicación apoyándonos en los principios de usabilidad de Nielsen.

Para ello iremos vista por vista en nuestra aplicación, después de hacer un resumen general sobre principios que se cumplan en todas y cada una de las vistas para así acortar el informe.

4.1. Usabilidad general

En la aplicación, podemos ver con facilidad que se cumplen varios principios de usabilidad de Nielsen.

Uno de ellos es *Consistencia y estándares*, ya que la totalidad de la aplicación tiene el mismo formato: fondo gris, tablas de fondo negro y letras blancas. También, todos los botones son con el mismo formato para botones de acción (azules) y los de cancelación (rojos). Cuando se pulsa en el logo de la aplicación, este nos envía a la vista base. Además, en todas las vistas vemos tenemos la barra superior para realizar dicha acción, a la vez que la inferior para poder ver información sobre los participantes del proyecto.

Otro principio de Nielsen que se cumple es el de *Diálogos estéticos y de diseño minimalista*. Este principio lo cumple, ya que las vistas de la aplicación no contienen información innecesaria que nos pueda entorpecer la comprensión de la información relevante al no existir ruido de información que puede provocar esta información extra. Además de tener un diseño minimalista, ya que las páginas no tienen gran número de colores, pocos elementos y mucho espacio, incluso en la vista base, la cual tiene texto para facilitar la comprensión de la aplicación.

Otro principio de Nielsen sería el de *Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores*. Cuando surge un error la aplicación envía una vista de error

instando al usuario a dirigirse a la vista base.

4.2. Login

4.3. Vistas de Médico

4.3.1. Listado de tareas

4.3.2. Asignación de tareas

4.4. Vistas de Técnico

4.4.1. Listado de robots

4.4.2. Detalles de los robots

4.4.3. Creación de tareas

Referencias

[1] *IFML: The Interaction Flow Modeling Language | The OMG standard for front-end design.* URL: https://www.ifml.org/.

[2] *UML Website.* URL: http://www.uml.org/.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA