Desarrollo de un escritorio web para la gestión de tareas

Memoria de Proyecto Final de Máster

Máster Universitario en Desarrollo de Sitios y Aplicaciones Web

Autor: Damián Serrano Thode

Profesor colaborador: Miguel Calvo Matalobos

Profesor: César Pablo Córcoles Briongos

01/01/2021

Copyright

Abstract

Este proyecto consiste en el desarrollo de un escritorio web para unificar en una única plataforma todas las tareas que actualmente se realizan de forma descoordinada y mediante diversos archivos desestructurados distribuidos en carpetas, que impiden la organización y la compartición de la información de forma automática. Adicionalmente se desarrollará la primera de las tareas que estarán incluídas en este escritorio que consiste en una aplicación web para la gestión de las llamadas estacionadas para recordar diversos eventos a los usuarios del servicio, como la toma de medicamentos, por ejemplo. En su desarrollo se utilizará Angular como framework web para el frontend de la aplicación y Typescript como lenguaje de programación, y .Net Core con el lenguaje C# para los servicios web de backend.

Palabras clave: gestión de tareas, angular, csharp, spa, aplicación web

Abstract (english version)

This project consists of the development of a web desktop for unifying in a single platform all the tasks that are performed in a disorganized way and via several unstructured files distributed in folders, which hinder information structuring and sharing on an automated way. Additionally, the first of such tasks will be developed, which consists on a web application for managing periodic calls to the users of the service to remind them of several events like taking medications, for example. Angular framework and the Typescript language will be used for the development of the frontend, and .Net Core with C# as the language for the development of the backend services.

Keywords: task management, angular, csharp, spa, web application

Índice

Índice de tablas

Índice de figuras

1. Introducción

Actualmente trabajo en una empresa pública que se dedica a la atención a las personas dependientes. Uno de las áreas de la empresa es un servicio de atención telefónica distribuido en dos centrales, que se encargan de la gestión de las llamadas de los usuarios del servicio de teleasistencia. Estos usuarios están distribuidos geográficamente entre ambas centrales, teniendo cada una de ellas repartidas las provincias que componen la comunidad autónoma.

Dentro del servicio de atención telefónica se llevan a cabo diversas tareas para la organización del trabajo, y estas tareas se distribuyen mediante archivos desestructurados a lo largo de una jerarquía de carpetas. Esto presenta una serie de problemas como la gestión del conocimiento acerca de la ejecución de estas tareas, la compartición de la información, la generación de informes sobre su progreso, el mantenimiento de dichos archivos, la gestión de accesos y seguridad sobre los mismos, por mencionar unos pocos.

Para solventar estas carencias se ha propuesto la creación de un entorno en el cual se puedan integrar las diferentes tareas, de modo que la información de estas se encuentre centralizada en una base de datos y se pueda facilitar la organización y la compartición de la información entre las distintas tareas, permitiendo la elaboración de informes automáticos, securización de los datos con diferentes perfiles de acceso, asignación automática de tareas a determinados grupos de personal según diversos parámetros, compartición de la información entre las diferentes plataformas tecnológicas existentes en la empresa, entre otras ventajas.

Además, se implementará la primera de estas tareas, consistente en la gestión de las llamadas periódicas a los usuarios del servicio. De forma periódica hay que realizar una serie de llamadas todos los días para recordar a los usuarios distintos eventos como la toma de medicamentos, por ejemplo. Actualmente, esta tarea se lleva a cabo entre ambas centrales telefónicas, en las cuales disponen de un listado de usuarios a los que llamar con los

motivos de la llamada, en una hoja de cálculo. La gestión de esta información se realiza por los supervisores del servicio de atención telefónica, los cuales van actualizando el archivo introduciendo nuevos registros, modificando los registros existentes o eliminándolos, según se van detectando las necesidades de los usuarios. Esta tarea se asigna de forma manual a un grupo de operadores telefónicos según el volumen de llamadas periódicas que haya que realizar, los cuales van progresando sobre el archivo línea a línea, realizando las llamadas en los tiempos indicados en el mismo.

En caso de incidencia técnica en alguna de las centrales, la tarea debe ser asignada a la otra central para lo cual deben comunicar al supervisor de la central en activo por qué línea del archivo de estacionadas se han quedado para que puedan continuar la tarea por dicho punto, y a la vuelta del servicio una vez solventada la incidencia técnica, comunicar hasta qué línea han avanzado en la otra central para retomar la actividad a partir de ese punto.

2. Descripción

Mediante este Trabajo Fin de Máster (TFM) se va a dar cobertura a las carencias expuestas en el apartado anterior a través de una plataforma web que unificará todas las tareas que se llevan a cabo en el servicio, y se implementará la primera de estas tareas consistente en la gestión de las llamadas estacionadas.

Una vez estudiadas las necesidades del servicio se ha propuesto la creación de un escritorio web que englobe todas las tareas como aplicaciones web independientes que se comunican a través de un conjunto de servicios web. Este escritorio contará con un conjunto de características principales que estarán disponibles para todas las aplicaciones:

- Gestión de permisos que permitan mostrar u ocultar aplicaciones según su nivel de acceso, así como configurar las aplicaciones según el nivel de acceso del usuario.
- Gestión de notificaciones al usuario, mediante elementos en pantalla o notificaciones nativas del sistema operativo.

Las aplicaciones incluidas en el escritorio web contarán con sus propios servicios de backend para la gestión de los datos, y estará disponible también un canal asíncrono de comunicación mediante mensajería para la distribución de notificaciones y eventos entre las distintas aplicaciones y el resto de plataformas tecnológicas de la empresa.

Y para dar uso a este escritorio web, se va a implementar la primera de las aplicaciones consistente en la gestión de las llamadas estacionadas. Esta aplicación tendrá las siguientes funcionalidades principales:

- Gestión de los elementos que forman parte de las llamadas estacionadas, permitiendo su inserción, modificación o eliminación.
- Búsquedas sobre los registros de la tarea según diferentes parámetros.
- Reparto de la tarea según el volumen de llamadas por intervalo y el número de operadores telefónicos disponible, así como asignación individual.

- Propuesta de nuevos registros a incluir, modificaciones o eliminaciones, por parte del personal operador telefónico.
- Generación de informes de rendimiento de la tarea.
- Coordinación de la tarea entre las distintas centrales que forman parte del servicio de atención telefónica.
- Notificación al usuario de las llamadas a realizar de forma inminente.
- Exportación de listados a archivos.

3. Objetivos

Los objetivos que se pretenden conseguir mediante la implementación de los proyectos descritos en este Trabajo Fin de Máster son los siguientes:

- Unificación de datos en una única plataforma centralizada.
- Estructuración y modelización de los datos que componen el núcleo de negocio del servicio.
- Comunicación de la información a lo largo de la estructura de la empresa.
- Gestión de los niveles de acceso para las tareas a realizar.
- Optimización de las tareas a realizar, permitiendo la automatización de tareas que sean susceptibles de ello.
- Permitir un presentación estandarizada y accesible a la información y a las tareas para los trabajadores con diversidad funcional.

4. Escenario

En el área de atención de llamadas se cuenta actualmente con una aplicación de escritorio en la cual se han desarrollado diversas funcionalidades para apoyo en el desempeño de la tarea diaria.

Esta aplicación esta desarrollada en C# y se encuentra desplegada en todos los puestos de trabajo, y si bien podría tomarse como punto de partida para el desarrollo de la nueva funcionalidad, se ha preferido realizar un nuevo desarrollo por los siguientes motivos:

- Esta aplicación es una aplicación de escritorio que presenta inconvenientes en cuanto al despliegue de la misma ya que para poder ofrecer una actualización a todos los usuarios es necesario desplegar la nueva versión de la aplicación en todos los puestos de trabajo, y aunque se cuenta con un sistema de despliegue automático, a veces se han encontrado problemas de permisos en el sistema de archivos para el uso de la aplicación entre todos los usuarios, o para aprovisionar un nuevo equipo de trabajo es necesario copiar dicha aplicación en el equipo.
- La aplicación se ha desarrollado en .Net Framework 3.5, el cual es una plataforma propietaria de Microsoft que solamente se puede ejecutar en el sistema operativo Windows. Actualmente se esta tendiendo al uso de plataformas abiertas y aplicaciones multiplataforma, por lo que el uso de dicha aplicación presenta inconvenientes en este sentido.
- El desarrollo de una plataforma web permite su uso desde cualquier equipo, independientemente de su sistema operativo, ya que el único requisito es que cuente con un navegador de internet moderno.
- La actualización de una aplicación web es inmediata, puesto que al desplegar en el servidor la nueva versión esta se encuentra disponible de forma inmediata para todos los usuarios.

A partir de los motivos indicados, se decidió descartar incluir la nueva funcionalidad en la aplicación de escritorio y desarrollar una plataforma web a la que, posteriormente, se le pueda incluir la funcionalidad existente en la

aplicación de escritorio, y de ese modo poder extinguir el uso de la misma, centrando toda la operativa en la nueva plataforma web.

5. Contenidos

La solución desarrollada en este trabajo de fin de master está compuesta por los siguientes elementos:

- Una aplicación web desarrollada en Angular que es el interfaz sobre el que el usuario va a actuar. En esta aplicación web se mostrará un escritorio en el cual tendrá disponibles una serie de módulos para realizar distintas tareas en la labor diaria de la atención de llamadas. Dentro del marco de este trabajo de fin de máster, además de desarrollar la estructura de este escritorio de aplicaciones, se ha desarrollado un módulo para la gestión de la tarea de las llamadas estacionadas.
- Un conjunto de servicios web desarrollados en .NET Core usando el lenguaje C# que dan acceso a la aplicación web a los datos del sistema y coordinan la gestión de los procesos. Además, estos servicios web se encargan de enviar notificaciones a través de un sistema de envío de mensajes para sincronizar la información entre todos los usuarios de la aplicación web.

6. Metodología

La metodología de desarrollo seguida en este proyecto ha sido una metodología ágil (1), dividida en dos fases bien diferenciadas.

En la primera fase se llevaron a cabo reuniones con al área funcional para determinar las necesidades y poder extraer un conjunto mínimo de funcionalidades a desarrollar. Una vez que se contaba con un esqueleto funcional básico se tomaron las decisiones que involucraban a la plataforma sobre la cual se debía ejecutar la aplicación y el entorno de desarrollo para la misma.

En la segunda fase se abarcó el desarrollo de la aplicación, estando en contacto estrecho con el área funcional para la revisión de las necesidades y el seguimiento del desarrollo. De este modo, se determinaron unos casos de uso y se estableció un ciclo de desarrollo y revisión por el área funcional para verificar que se habían cumplido los objetivos funcionales de cada caso de uso.

7. Arquitectura

7.1. El modelo C4 de visualización de arquitecturas software

Para la descripción de la arquitectura se va a usar el modelo C4 para la visualización de arquitecturas software (2). Este modelo fue creado por el arquitecto de software Simon Brown, y toma como orígenes el Lenguaje Unificado de Modelado (UML) (3) y el modelo de visualización de arquitecturas 4+1 (4).

La necesidad de este sistema de visualización parte a raíz de estandarizar la presentación de un sistema de software más allá de utilizar *cajas y líneas* sin ningún criterio, pero a la vez proporcionando un sistema de representación más ágil que la notación UML, ArchiMate o SysML.

Según este modelo, se utiliza una notación gráfica para modelar la arquitectura de los sistemas software. Está basado en la descomposición estructural de un sistema en contenedores y componentes, y utiliza la notación basada en UML para la descomposición más detallada de los bloques que conforman la arquitectura.

La presentación de la arquitectura se basa en cuatro capas jerárquicas en las cuales se muestra el sistema software cada vez con mayor detalle, siendo cada una de las capas las que se indican a continuación:

- Nivel 1: Un diagrama de contexto del sistema que proporciona un punto de inicio, y muestra el modo en el que el sistema software encaja con el resto de sistemas a su alrededor.
- Nivel 2: Un diagrama de contenedores, que descompone el sistema software en contenedores relacionados, y a su vez sus relaciones con los sistemas software a su alrededor.
- Nivel 3: Un diagrama de componentes, que profundiza en un contenedor y muestra los componentes que lo forman, y a su vez la relación de éstos con los sistemas software externos.

- Nivel 4: Un diagrama de código (habitualmente un diagrama de clase UML), que muestra la implementación de un componente.

Mediante este sistema de visualización se proporcionan detalles de la arquitectura para cada nivel de consumo de la misma, puesto que no todo el personal involucrado en un proyecto necesita el mismo nivel de detalle de la misma.

7.2. Arquitectura del escritorio de aplicaciones

A continuación se va a detallar la arquitectura del escritorio de aplicaciones utilizando el modelo C4 para la visualización de arquitecturas software. Mediante este sistema, se mostrará esta arquitectura en varias capas, incrementando el nivel de detalle en cada una de ellas.

En una primera capa se mostrará una visión general de la arquitectura de la aplicación junto con el resto de sistemas software que existen a su alrededor. En una segunda capa se profundizará en la aplicación y se mostrará cada uno de los contenedores que la componen. En una tercera capa, se detallará la composición de cada uno de los contenedores en sus distintos componentes, y en una cuarta y última capa se mostrará un diagrama detallado de la estructura de cada uno de los componentes.

7.2.1. Nivel 1: Contexto del sistema

En esta capa se muestra la ubicación de la aplicación del Escritorio Web junto con el resto de sistemas externos con los que se comunica. En la figura 1 se puede observar el modo en el que la aplicación del escritorio web encaja con el resto de sistemas a su alrededor.

En la figura se muestran los distintos componentes del sistema para la vista de alto nivel. Por un lado se muestra el usuario y el sistema software del escritorio web, que son los componentes principales, y por otro lado se muestran los componentes externos al sistema, que son la base de datos existente y una serie de servicios web existentes que proporcionan funciones

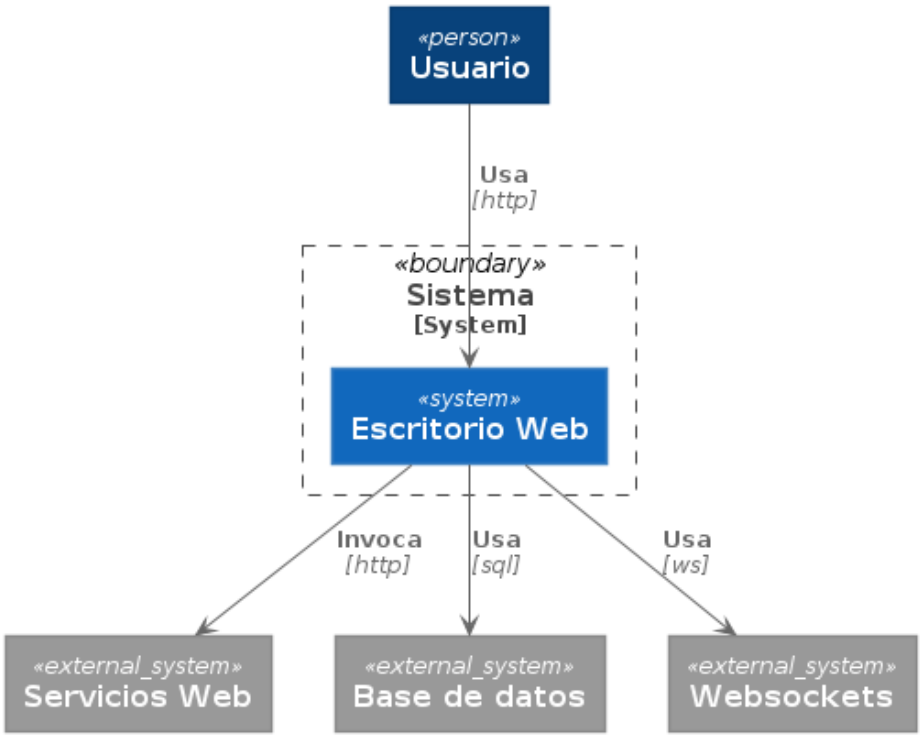

Figura 1: Diagrama de contexto del sistema

auxiliares como por ejemplo autenticación o auditoría.

7.2.2. Nivel 2: Diagrama de contenedores

En esta capa de diagramas se muestra la composición de subsistemas de la aplicación, mostrando los bloques a alto nivel.

En la figura 2 se puede observar la composición de elementos del sistema software del escritorio web, englobado en la línea discontinua.

Por un lado se encuentra el contenedor del servidor web que ofrece el contenido de la aplicación, tanto de archivos estáticos HTML, JavaScript y CSS, como imágenes.

Por otro lado se encuentra el contenedor del escritorio web propiamente, el cual ofrece la aplicación que se ejecuta en el navegador del usuario. Esta aplicación ofrece toda la funcionalidad con la que va a interactuar el usuario y es la capa visible al usuario de todo el sistema.

A continuación se encuentra la API de servicios, la cual ofrece el procesamien-

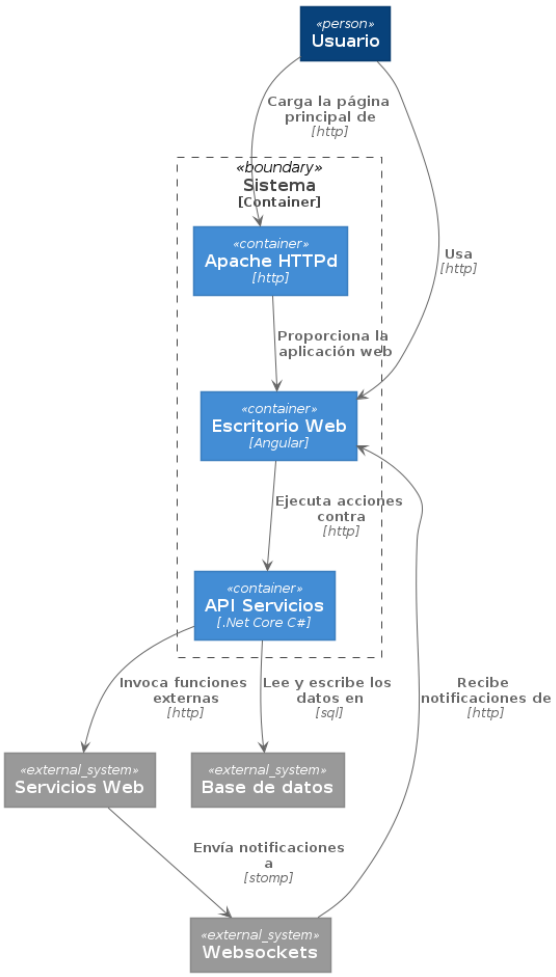

Figura 2: Diagrama de contenedores del sistema

to de las operaciones realizadas por el usuario en la aplicación web. Esta API se comunica con la base de datos para obtener la información y gestionar los datos, y también se comunica con los sistemas externos a la aplicación como son los servicios web que ofrecen las funcionalidades de autenticación o auditoría. La API de servicios también se comunica con el servidor de Websockets para enviar las notificaciones de actualizaciones de los datos, de modo que la aplicación web pueda recibir estas actualizaciones y reflejar los cambios en el interfaz. Estas actualizaciones podrán venir tanto de las acciones realizadas por el propio usuario, como de las acciones realizadas por cualquier otro usuario de la aplicación web.

7.2.3. Nivel 3: Diagrama de componentes

En esta capa se muestran los componentes que forman la estructura de los contenedores de la capa anterior.

Para este nivel se van a mostrar los componentes que forman parte tanto del contenedor de la aplicación web como del contenedor de la API de servicios.

7.2.3.1. Componentes del escritorio web La aplicación web se ha diseñado basándose en el patrón de desarrollo Redux XXXcita, mediante el cual, el estado de la aplicación se encuentra en un almacenamiento centralizado y todos los elementos de la aplicación leen el estado de dicho almacenamiento y lanzan acciones que modificarán el estado, pero sin manipular directamente el estado global.

Los componentes que forman parte del escritorio web son los que se muestran en la figura 3. En ella se pueden observar los siguientes elementos:

Componente raíz: Este componente es la raíz de la aplicación web y dentro de él se añadirán los distintos componentes que forman parte del global de la aplicación.

main-desktop: Este componente contiene la lista de aplicaciones registradas en el sistema para las que el usuario tiene permiso de acceso.

main-menu: Este componente contiene la barra superior de la aplicación en la que se muestra el botón de vuelta al escritorio, el título del módulo activo y el botón para acceder a las notificaciones.

main-notificaciones: Este componente mostrará el icono con un indicador de las notificaciones recibidas del sistema.

main-notificaciones-container: Este componente tiene la lista de notificaciones y se mostrará cuando el usuario haga click sobre el componente anterior del icono de notificaciones.

authentication: Este componente hace de punto de entrada de la aplicación para autenticar a los usuarios.

no_auth: Este componente es el destino final si ha habido algún problema con la autenticación de los usuarios.

logout: Este componente es el destino final cuando el usuario ha salido del sistema.

ApplicationRegistryService: Este servicio se encarga de registrar cada una de las aplicaciones incluidas en la plataforma. Para cada una de las aplicaciones que se quieran registrar validará si el usuario tiene permisos para acceder a la misma y, si los tiene se añadirá la aplicación, y si no los tiene se descartará y la aplicación no estará disponible para dicho usuario.

Servicio de notificaciones: Este componente se encarga de interactuar con el servidor de Websockets para gestionar las notificaciones recibidas del sistema y trasladar al usuario las notificaciones que estén relacionadas con su actividad.

Servicio de usuarios: Este servicio se encarga de consultar en la API los datos de los usuarios del sistema.

Servicio de autenticación: Este servicio se encarga de realizar al autenticación de los usuarios.

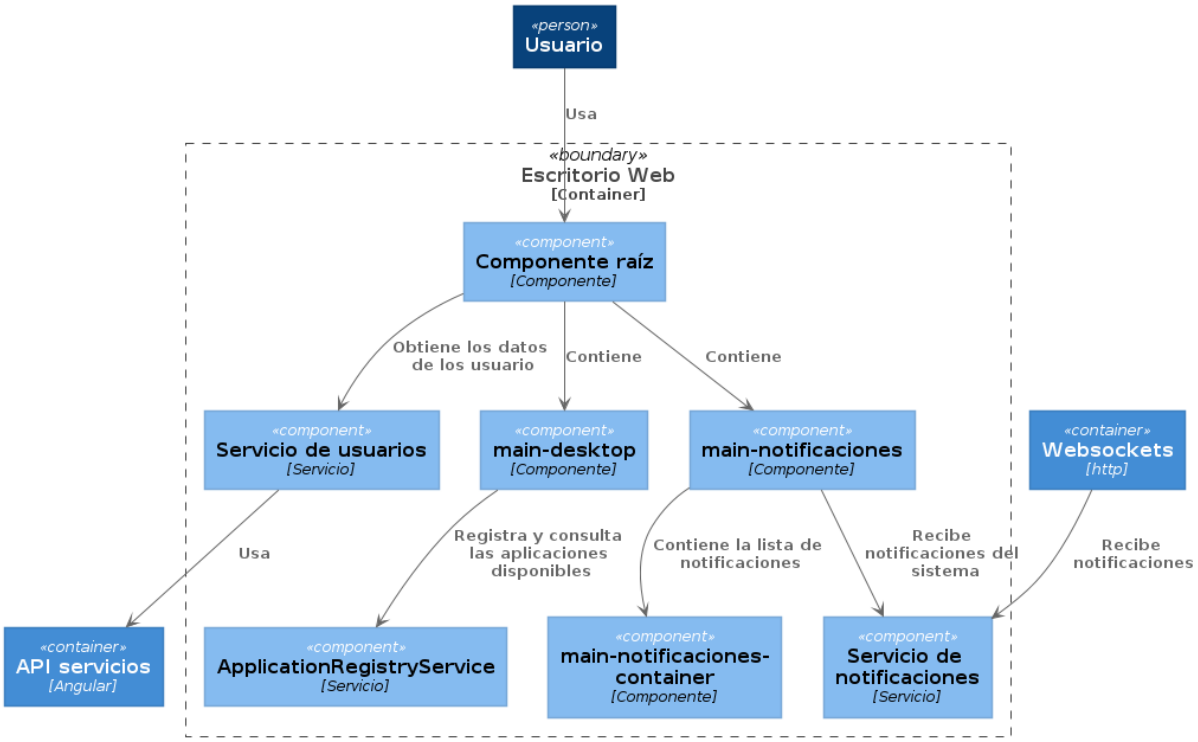

Figura 3: Diagrama de componentes de la aplicación web

7.2.3.2. Componentes de la API de servicios Los componentes que forman parte de la API de servicios son los que se muestran en la figura 4.

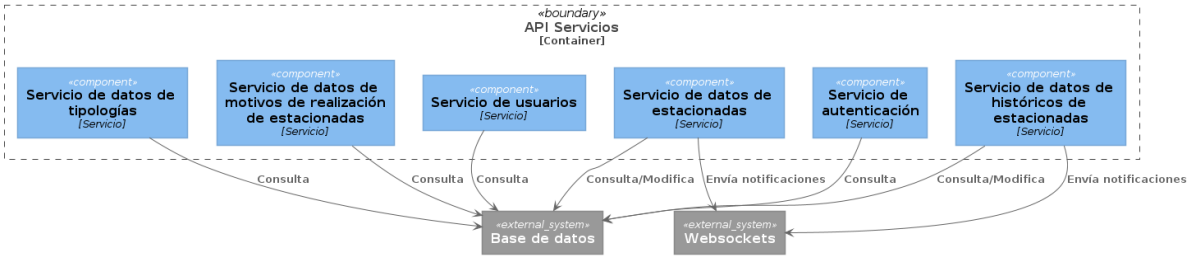

Figura 4: Diagrama de componentes de la API de servicios

La descripción de cada uno de los siguientes elementos se muestra a continuación:

Servicio de autenticación: Este servicio se encarga de autenticar a los usuarios para permitir el acceso al sistema. En caso de tener éxito en la autenticación devuelve los datos del usuario y un token de sesión que se debe enviar en cada posterior consulta a los servicios web.

Servicio de usuarios: Este servicio se encarga de ofrecer la información sobre los usuarios del sistema.

Servicio de datos de estacionadas: Este servicio de encarga de proporcionar los datos de llamadas estacionadas y realizar las modificaciones solicitadas por los usuarios.

Servicio de datos de históricos de estacionadas: Este servicio se encarga de crear los registros de históricos de llamadas estacionadas cuando se realizan las llamadas.

Servicio de datos de motivos de realización de estacionadas: Este servicio se encarga de proporcionar los datos de los motivos de realización de llamadas estacionadas, usados en la creación de los históricos de estacionadas.

Servicio de datos de tipologías: Este servicio de encarga de proporcionar los datos de las tipologías de las llamadas para los filtros.

7.2.3.3. Componentes del módulo de llamadas estacionadas El módulo de llamadas estacionadas, al estar incluido dentro de la aplicación de Escritorio SAT, también hace uso del patrón de desarrollo Redux, y el estado de dicho módulo se ha añadido al objeto de estado global de la aplicación.

Los componentes que forman parte del módulo de llamadas estacionadas son los que se muestran en la figura 5. En dicho diagrama se pueden observar los siguientes elementos:

estacionadas: Este es el componente raíz del módulo y contiene el componente de la tabla de llamadas estacionadas.

tabla-estacionadas: Este componente es el que tiene la tabla de llamadas estacionadas y el resto de componentes que forman parte del aspecto visual de la aplicación.

botonera-acciones-principal: Este componente contiene la botonera para las acciones que se pueden realizar en el módulo como añadir una nueva llamada estacionada, editar una llamada estacionada seleccionada o eliminar una llamada estacionada seleccionada.

detalle-fila: Este componente muestra el detalle de los datos de una llamada

estacionada seleccionada en la tabla.

filtros-estacionadas: Este componente muestra los campos de filtrado para la tabla de datos de llamadas estacionadas. Estos campos permitirán filtrar por cada una de las columnas de la tabla.

fila-estacionadas: Este componente contiene una fila de datos de la tabla de llamadas estacionadas.

formulario-estacionada: Este componente muestra el formulario de una llamada estacionada, y es utilizado tanto para añadir una nueva llamada estacionada como para editar un registro existente.

formulario-asignacion: Este componente muestra el formulario para asignar la llamada estacionada a un usuario del sistema.

formulario-realizacion: Este componente muestra el formulario para seleccionar el motivo de realización de la llamada estacionada para posteriormente crear el registro histórico.

Servicio de estacionadas: Este servicio accede a la API para obtener y manipular los datos de estacionadas.

Servicio de motivos de estacionadas: Este servicio accede a la API para obtener los datos de motivos de realización de llamadas estacionadas.

Servicio de histórico de estacionadas: Este servicio accede a la API para crear los registros de históricos de realización de llamadas estacionadas.

Servicio de tipologías: Este servicio accede a la API para obtener los registros de tipologías de llamadas estacionadas.

7.2.3.4. Gestión de datos en el Escritorio SAT

Para toda la aplicación se ha organizado la gestión de los datos utilizando el patrón Redux, como se ha mencionado anteriormente.

Este patrón permite organizar los datos de forma centralizada de modo que únicamente existe una única *fuente de verdad* de datos para toda la aplicación, de la que el resto de componentes puede leer, pero no modificar

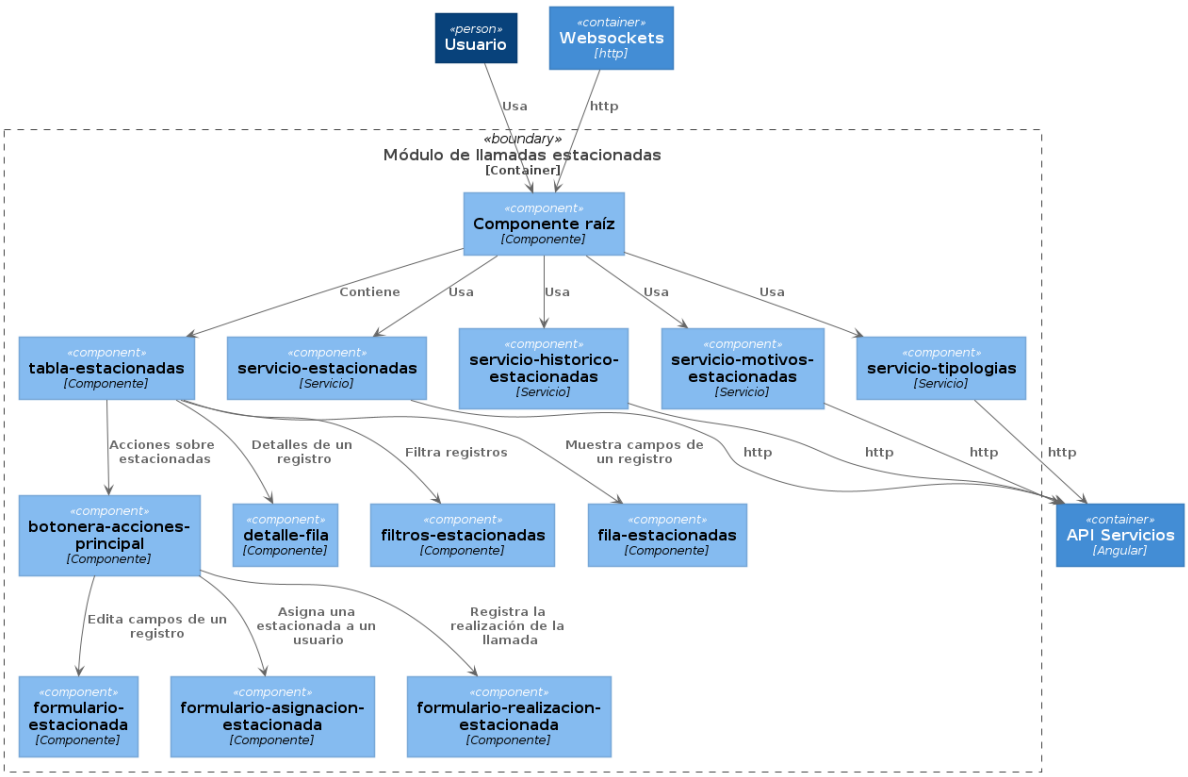

Figura 5: Diagrama de componentes del módulo de llamadas estacionadas

directamente.

Para modificar los datos se realizan unas acciones que pueden tener efectos sobre el repositorio de datos modificando los mismos, pero de forma centralizada.

Las acciones creadas para el componente del Escritorio SAT son las siguientes:

add-notification: Esta acción añade una notificación al estado del sistema, para posteriormente ser mostrada al usuario en su componente correspondiente.

authenticate-user-error: Esta acción refleja el error en la autenticación de un usuario.

authenticate-user-ok: Esta acción refleja el éxito en la autenticación de un usuario.

autenthicate-user: Esta acción refleja la intención de realizar la autenticación del usuario.

load-users-error: Esta acción refleja un error en la carga de la lista de usuarios.

load-users-ok: Esta acción refleja el éxito en la carga de la lista de usuarios.

load-users: Esta acción refleja la intención de cargar la lista de usuarios.

logout-user: Esta acción refleja la intención de cerrar la sesión del usuario.

open-application-ok: Esta acción refleja el éxito en la pertura de una aplicación.

open-application: Esta acción refleja la intención de abrir una aplicación.

open-desktop-ok: Esta acción refleja el éxito en la apertura del escritorio.

open-desktop: Esta acción refleja la intención de abrir el escritorio.

read-notification: Esta acción refleja el hecho de leer una notificación pendiente.

register-application-ok: Esta acción refleja el éxito en el registro de una aplicación en el catálogo de aplicaciones.

register-application: Esta acción refleja la intención de registrar una aplicación en el catálogo.

toggle-notificaciones-container: Esta acción refleja el hecho de mostrar u ocultar el contenedor de notificaciones.

Para estas acciones se han configurado los siguientes efectos, que son los componentes de la aplicación que se encargan de interactuar con sistemas externos:

authenticate-user: Invoca la API de autenticación para validar los datos del usuario.

authenticate-user-ok: Recibe de la API de autenticación los datos del usuario y los incorpora al servicio de autenticación.

authenticate-user-error: Redirige al usuario al componente de error donde se le muestra un mensaje de error en la autenticación.

logoutUser: Vacía los datos de sesión del servicio de autenticación y redirige al usuario al componente de cierre de sesión.

registerApplication: Comprueba los datos de registro del módulo que se está intentando cargar y carga las rutas del módulo en el componente de rutas de la aplicación.

openApplication: Hace la navegación al componente principal del módulo que se está intentando abrir.

openDesktop: Hace la navegación al componente de escritorio.

loadUsers: Invoca a la API de usuarios para obtener el listado.

Y por último, para completar los componentes de la gestión de estados se han definido los siguientes *reducers* que son los que manipulan el estado global de la aplicación:

registerApplicationOk: Añade la aplicación al catálogo de aplicaciones.

openApplicationOk: establece el valor de la aplicación actual.

openDesktopOk: Establece el valor nulo en la aplicación actual.

toggleNotificacionesContainer: Intercambia el valor de la variable que indica si el contenedor de notificaciones se debe mostrar o no.

loadUsersOk: Establece el valor de la lista de usuarios.

addNotification: Añade una nueva notificación a la lista.

readNotification: Establece el parámetro “leído” de la notificación que se indica por su posición en la lista.

7.2.3.5. Gestión de datos en el módulo de estacionadas Al igual que en la aplicación del Escritorio SAT, el módulo de llamadas estacionadas también gestiona su estado a través del patrón Redux, y este se encuentra incorporado al estado global de la aplicación.

Las acciones que se han definido para el módulo de estacionadas son las siguientes:

actualizar-estacionada-error: Ha ocurrido un error al actualizar los datos de un registro de llamada estacionada.

actualizar-estacionada-ok: Se ha modificado correctamente un registro de llamada estacionada.

actualizar-estacionada: Indica la intención de modificar un registro de llamada estacionada.

cargar-estacionadas-error: Ha ocurrido un error al cargar la lista de llamadas estacionadas.

cargar-estacionadas-ok: La lista de llamadas estacionadas se ha cargado correctamente.

cargar-estacionadas: Indica la intención de cargar la lista de llamadas estacionadas.

cargar-estacionadas-motivos-error: Ha ocurrido un error al cargar los motivos de realización.

cargar-estacionadas-motivos-ok: Se ha cargado correctamente la lista de motivos de realización.

cargar-estacionadas-motivos: Indica la intención de cargar los motivos de realización.

cargar-tipologias-error: Ha ocurrido un error al cargar la lista de tipologías.

cargar-tipologias-ok: Se ha cargado correctamente la lista de tipologías.

cargar-tipologias: Indica la intención de cargar la lista de tipologías.

crear-estacionada-error: Ha ocurrido un error al crear un registro de llamada estacionada.

crear-estacionada-ok: Se ha creado correctamente el registro de llamada estacionada.

crear-estacionada: Indica la intención de crear un registro de llamada estacionada.

eliminar-estacionada-error: Ha ocurrido un error al eliminar un registro de llamada estacionada.

eliminar-estacionada-ok: Se ha eliminado correctamente un registro de llamada estacionada.

eliminar-estacionada: Indica la intención de eliminar un registro de llamada estacionada.

mensaje-actualizacion-estacionada: Se ha recibido un mensaje de actualización de un registro de llamada estacionada a través de Websockets.

mensaje-eliminacion-estacionada: Se ha recibido un mensaje de eliminación de un registro de llamada estacionada a través de Websockets.

mensaje-insercion-estacionada: Se ha recibido un mensaje de inserción de un registro de llamada estacionada a través de Websockets.

realizar-estacionada: Indica la intención de crear un registro de histórico de realización de llamada estacionada.

seleccionar-estacionada: Se ha pulsado sobre una fila en la tabla de llamadas estacionadas.

Para estas acciones se han configurado los siguientes efectos en el módulo de llamadas estacionadas:

cargarEstacionadas: Invoca a la API de servicios para obtener la lista de llamadas estacionadas.

cargarTipologias: Invoca a la API de servicios para obtener la lista de tipologías.

crearEstacionada: Invoca a la API de servicios para crear un registro de llamada estacionada.

actualizarEstacionada: Invoca a la API de servicios para actualizar un registro de llamada estacionada.

eliminarEstacionada: Invoca a la API de servicios para eliminar un registro de llamada estacionada.

cargarEstacionadaMotivos: Invoca a la API de servicios para cargar la lista de motivos de realización de llamadas estacionadas.

realizarEstacionada: Invoca a la API para crear el registro de histórico de realización de llamada estacionada.

Y por último, los *reducers* que se han creado en el módulo de llamadas estacionadas son los siguientes:

cargarEstacionadasOk: Incorpora al estado de la aplicación la lista de llamadas estacionadas obtenida de la API.

seleccionarEstacionada: Establece la fila seleccionada como la fila que se ha pulsado en la tabla.

cargarTipologiasOk: Incorpora al estado de la aplicación la lista de tipologías obtenida de la API.

mensajeInsercionEstacionada: Incorpora a la lista de llamadas estacionadas el registro recibido a través de Websockets.

mensajeActualizacionEstacionada: Actualiza el registro de llamada estacionada con los datos recibidos a través de Websockets.

mensajeEliminacionEstacionada: Elimina de la lista el registro recibido a través de Websockets.

cargarEstacionadasMotivosOk: Incorpora al estado global la lista de motivos de realización de llamadas estacionadas obtenida a través de la API.

7.3. Arquitectura de los servicios web

En este apartado se va a detallar la arquitectura de los servicios web que forman la API de servicios para la aplicación web de Escritorio SAT, usando el modelo C4 de visualización de arquitecturas software al igual que el apartado anterior.

7.3.1. Nivel 1: Contexto del sistema

En este nivel se presenta la ubicación de los servicios web mediante un diagrama generalista que muestra los componentes a un alto nivel.

En la figura 6 se puede observar dicho diagrama y cómo encajan estos servicios web en la arquitectura general de la solución.

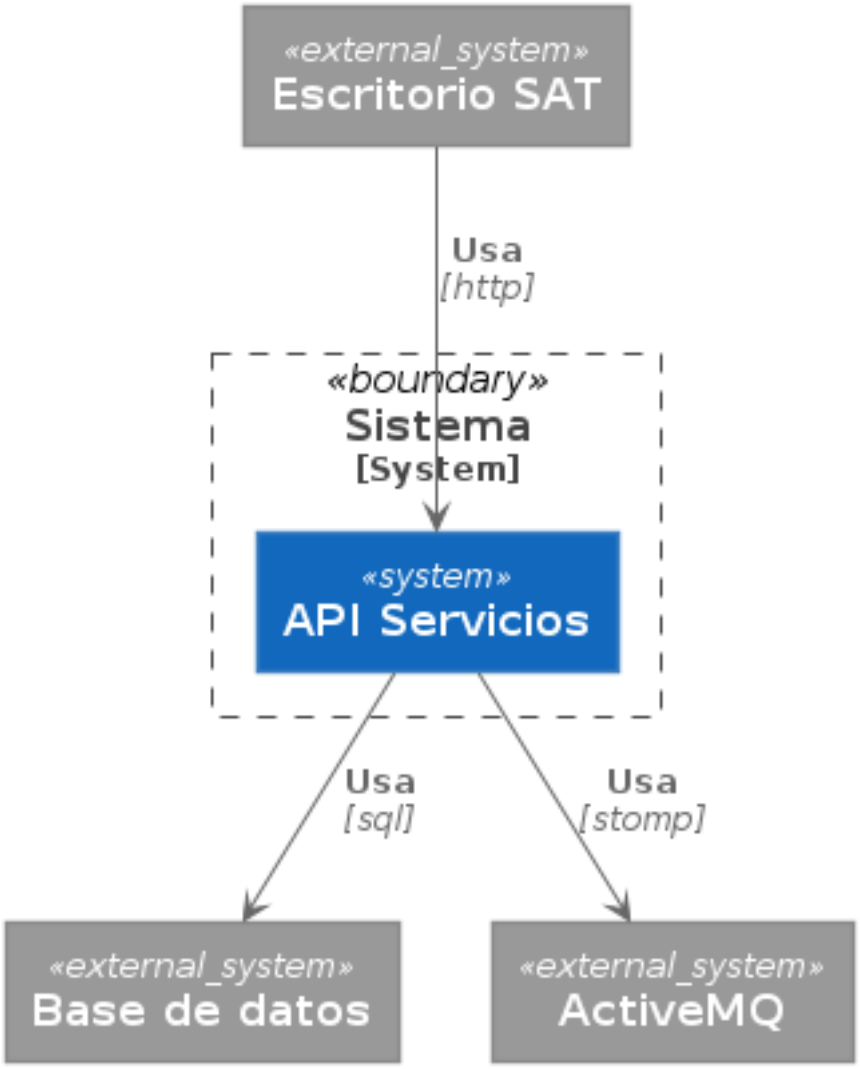

Figura 6: Diagrama de contexto de la API de servicios web

7.3.2. Nivel 2: Diagrama de contenedores

En este nivel de visualización de la arquitectura de los servicios web que forman la API de servicios para el Escritorio SAT, se muestra la composición de subsistemas de los mismos, mostrando los bloques a alto nivel.

En la figura 7 se puede ver la organización de los componentes dentro del sistema de servicios web de la API, englobados por una línea discontinua.

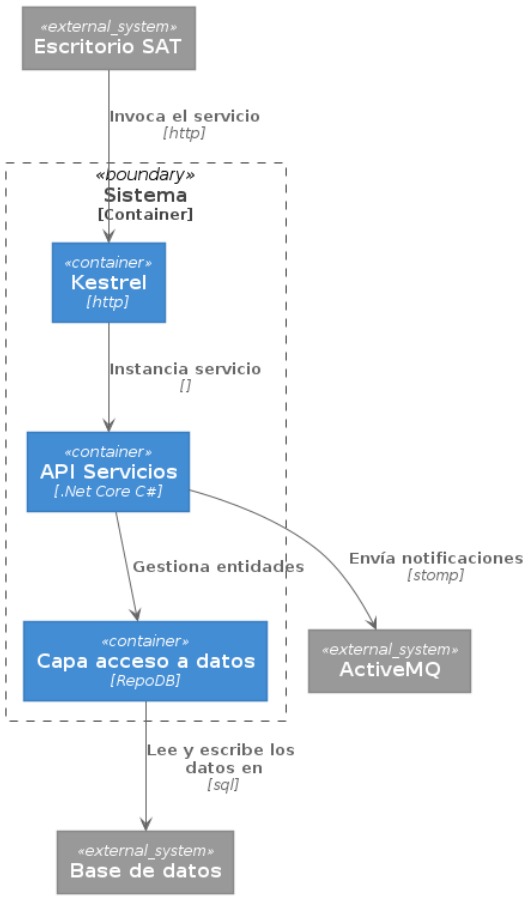

Figura 7: Diagrama de contenedores de la API de servicios web

En la figura se puede observar la estructura de los contenedores del sistema. En ésta se muestra el contenedor HTTP Kestrel de .NET Core que hace de fachada para las peticiones HTTP al servicio, el cual redirige las peticiones a una instancia del servicio web realizado en C#.

Para el acceso a los datos, el servicio utiliza una capa de abstracción usando la librería RepoDB para el acceso a la base de datos Microsoft SQL Server, y posteriormente las notificaciones sobre las acciones en el sistema son enviadas a través del servicio de mensajería ActiveMQ usando el protocolo STOMP.

7.3.3. Nivel 3: Diagrama de componentes

En este nivel de visualización se muestra la estructura de un servicio web al detalle, mostrando cada una de sus partes. En la figura 8 se puede observar la composición del servicio web de llamadas estacionadas.

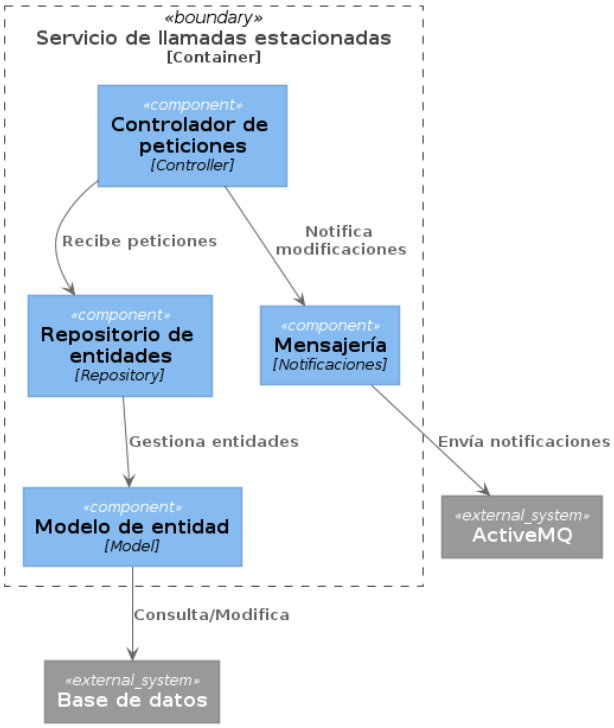

Figura 8: Diagrama de componentes del servicio web de llamadas estacionadas

En dicha figura se puede observar que el servicio contiene los siguientes componentes:

Controlador: Este componente se encarga de recibir la petición y orquestar el procesamiento de la misma.

Repositorio: Este componente se encarga de gestionar las entidades de datos del servicio.

Modelo: Este componente representa entidades concretas de la base de datos.

Base de datos: Este componente externo almacena los datos de la aplicación.

ActiveMQ: Este componente externo se utiliza para sincronizar las notificaciones entre todos los usuarios de la aplicación. El servidor publica notificaciones y todos los usuarios de la aplicación web las reciben.

La estructura del resto de servicios web que forman parte de la API de servicios es similar a la mostrada en el diagrama de este servicio de llamadas estacionadas.

8. Plataforma de desarrollo

Las tecnologías que se van a usar para implementar este Trabajo Fin de Máster son las siguientes:

- Angular para el framework de frontend tanto del escritorio web como de las aplicaciones que englobará.
- Angular Material para el aspecto común de los componentes del frontend.
- .Net Core con C# para las APIs de servicios de backend.
- SQL Server como gestor de base de datos y RepoDB (5) como librería para el acceso a los datos.
- Websockets para la comunicación asíncrona entre los distintos elementos de las aplicaciones y los distintos usuarios del sistema. Como sistema para la gestión de la mensajería se ha usado el software Apache ActiveMQ, el cual ofrece interfaz mediante el protocolo STOMP (6) para ser usado por los servicios web en C# y un interfaz de Websockets para ser usado por la aplicación web en Angular.
- Docker y Docker Compose para la orquestación de los distintos componentes de la aplicación.

9. Planificación

De acuerdo a las historias de usuario descritas en el anexo 2, se ha elaborado la siguiente planificación para el desarrollo del TFM que se puede observar en la tabla 1.

Se ha organizado el desarrollo en una serie de sprints que se ejecutarán de forma secuencial, y cada sprint estará englobado por las tareas descritas a continuación.

El reparto de tareas se ha estimado con una carga distribuida homgéneamente en cada sprint y se irá revisando según avance el desarrollo, comprobando la velocidad de desarrollo del proyecto.

Tabla 1: Estimación de la planificación de desarrollo

Sprint	Tareas	Desde	Hasta
	Definición del TFM	17/02/2021	26/02/2021
	Elaboración inicial de la memoria	27/02/2021	02/03/2021
	Entrega PEC1	02/03/2021	02/03/2021
Sprint 1	HUEWC01, HUEWC02, HUEWC03	03/03/2021	09/03/2021
Sprint 2	HUEWU01, HUEWU02, HUEWU03	10/03/2021	16/03/2021
Sprint 3	HUEWU04, HUEWU05, HULLC01	17/03/2021	23/03/2021
Sprint 4	HULLC02	24/03/2021	28/03/2021
	Revisión de la memoria	29/03/2021	31/03/2021
	Entrega PEC2	31/03/2021	31/03/2021
Sprint 5	HULLS01, HULLS02, HULLS03	01/04/2021	07/04/2021
Sprint 6	HULLS04, HULLS05, HULLS06	08/04/2021	14/04/2021
Sprint 7	HULLS07, HULLS08, HULLS09	15/03/2021	21/04/2021
Sprint 8	HULLOP01, HULLOP02, HULLOP03	22/04/2021	28/04/2021
Sprint 9	HULLOP04, HULLOP05, HULLOP06	29/04/2021	05/05/2021
	Revisión de la memoria	06/05/2021	09/05/2021
	Entrega PEC3	09/05/2021	09/05/2021

Sprint	Tareas	Desde	Hasta
Sprint 10	HULLOP07, HULLOP08, HULLOP09	10/05/2021	16/05/2021
Sprint 11	HULLOP10, HULLS10, HULLS11	17/05/2021	23/05/2021
Sprint 12	HULLS12, HULLS13	24/05/2021	30/05/2021
	Revisión de la memoria	31/05/2021	03/06/2021
	Elaboración de la presentación	04/06/2021	07/06/2021
	Entrega final	07/06/2021	07/06/2021

En la figura 9 se puede observar un diagrama de Gantt con la planificación completa del TFM.

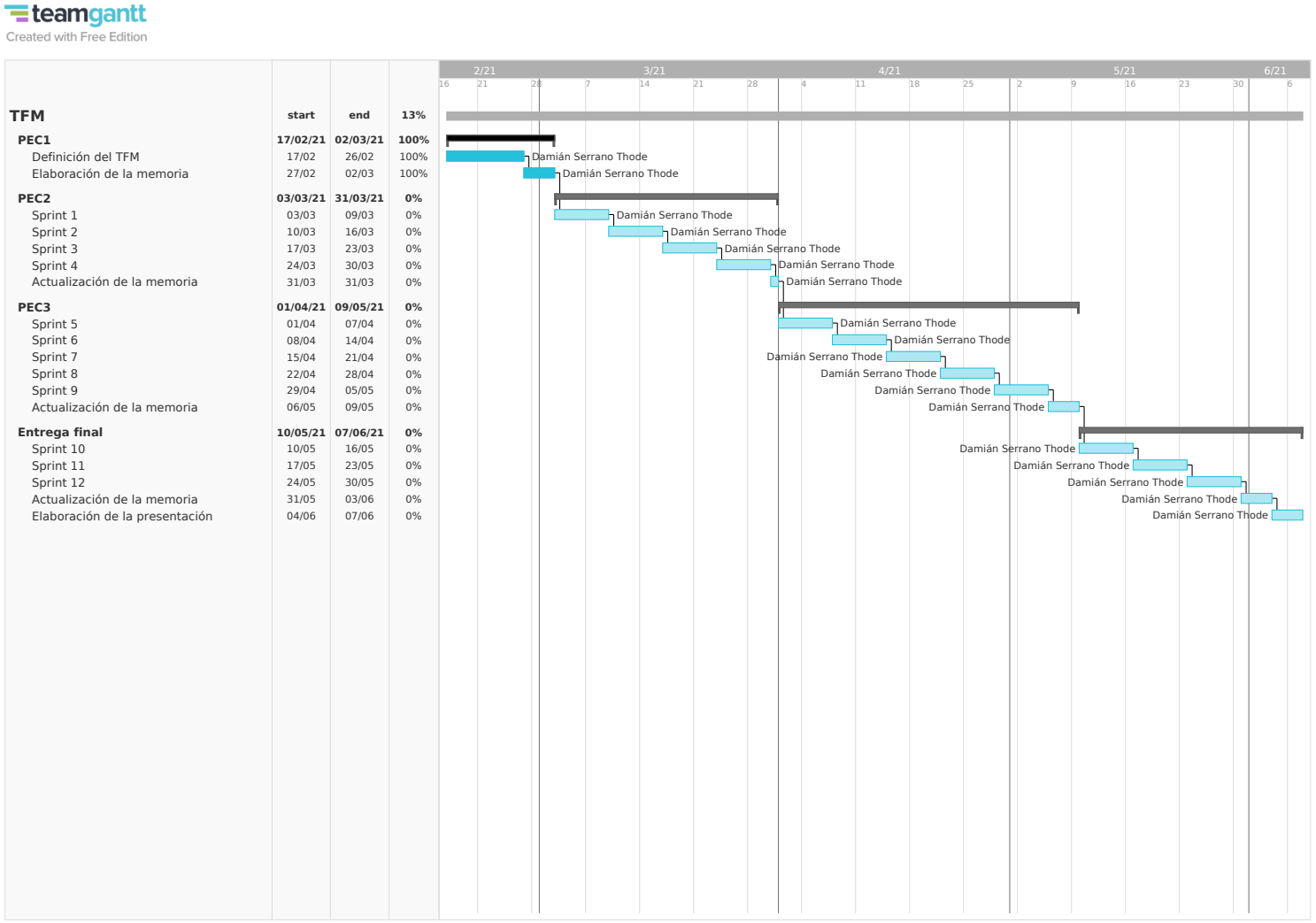

Figura 9: Diagrama de Gantt del proceso de desarrollo

9.1. Resultado de la planificación

De acuerdo a la planificación inicial, ha habido una desviación en la cantidad de tareas que se han podido llevar a cabo. La planificación partía de una propuesta bastante ambiciosa y cuando se ha ido abordando el desarrollo de cada uno de los ítems, se han encontrado complicaciones que han causado retrasos en la misma.

Por lo tanto, según avanzaba el desarrollo de la solución, en coordinación con la parte funcional del proyecto, se determinaron una serie de tareas que eran de menor valor funcional para ser realizadas en una segunda fase del proyecto. Las tareas son las siguientes:

- HULLOP06, esta tarea permite al personal operador introducir nuevos registros en la lista de llamadas estacionadas que posteriormente serán revisados por el personal supervisor para decidir su inclusión en el listado o no.
- HULLOP07, esta tarea permite al personal operador proponer modificaciones a los registros de llamadas estacionadas que posteriormente serán revisados por el personal supervisor para decidir su aplicación o no.
- HULLOP08, esta tarea permite al personal operador sugerir eliminaciones de registros de llamadas estacionadas que posteriormente serán revisados por el personal supervisor para decidir su aplicación o no.
- HULLS06, esta tarea permite al personal supervisor revisar la carga de llamadas planificada para el día. Aunque no se ha implementado un resumen del volumen de llamadas planificadas para el día, la lista completa se puede revisar en la aplicación, por lo que la funcionalidad estaría parcialmente cubierta.
- HULLS10, esta tarea permite al personal supervisor obtener un informe de las llamadas estacionadas realizadas en el turno. Esta funcionalidad no se ha implementado en la aplicación, pero el sistema de atención de llamadas permite extraer listados a partir de los motivos de llamada,

por lo que la funcionalidad estaría parcialmente cubierta.

- HULLS11, esta funcionalidad permite a los supervisores revisar las sugerencias de inserciones, modificaciones o eliminaciones de registros de llamadas estacionadas por parte de los teleasistentes, pero al haber aplazado el desarrollo de dichas tareas, esta funcionalidad no tiene sentido implementarla.
- HULLS12, esta funcionalidad permite al personal supervisor exportar a un archivo el listado que se esta visualizando en la aplicación.

Llegado a la finalización del trabajo de fin de máster, se han implementado un total de 26 tareas, del global de 33 tareas planificadas.

Según el ritmo de desarrollo llevado durante el proyecto, para estas 6 tareas que han quedado pendientes se estima un plazo de ejecución de entre dos a tres semanas.

10. Proceso de trabajo

Como se ha visto en el capítulo anterior, el proceso de trabajo de este proyecto ha consistido en planificar el desarrollo según las fechas de cada entrega.

El conjunto de funcionalidades se dividió en sprints que fueron realizados de acuerdo con las fechas de las entregas parciales del trabajo de fin de máster.

11. Prototipos

A través del programa Figma se crearon prototipos del aspecto de la aplicación, con interacciones sencillas para mostrar una aproximación del comportamiento de la misma.

En las siguientes secciones se va a mostrar el prototipo de cada una de las pantallas de la aplicación.

11.1. Escritorio

Esta pantalla es la pantalla principal que se carga cuando un usuario inicia la aplicación. En la figura 10 se puede observar el aspecto de la misma.

En dicha pantalla se muestra una barra superior que contiene los siguientes elementos:

- Un botón para volver al escritorio, que en la pantalla de escritorio está inactivo ya que se encuentra en la misma pantalla que su destino.
- Un título que indica la aplicación en la que se encuentra el usuario.
- Un botón con el icono de una campana para mostrar las notificaciones del usuario. Cuando se pulse, se mostrará un contenedor con la lista de notificaciones que tenga el usuario en este momento.

A continuación se encuentra el espacio para la lista de aplicaciones. Esta lista mostrará todas las aplicaciones para las que tiene acceso el usuario, con el siguiente contenido para cada una:

- El nombre de la aplicación.
- Una breve descripción de su funcionalidad.
- Un botón que abrirá la aplicación al pulsarlo.

Cuando el usuario pulse sobre el icono de la campana de notificaciones, se mostrará un contenedor con la lista de notificaciones. El aspecto de dicho contenedor se puede observar en la figura 11, y la descripción de su contenido es la siguiente:

- Un botón de cierre del contenedor.

- Una barra de título que indica que se trata del contenedor de notificaciones.
- Una lista de notificaciones, para las cuales se mostrará un icono que indique el tipo de notificación, y un breve texto del contenido de la notificación.

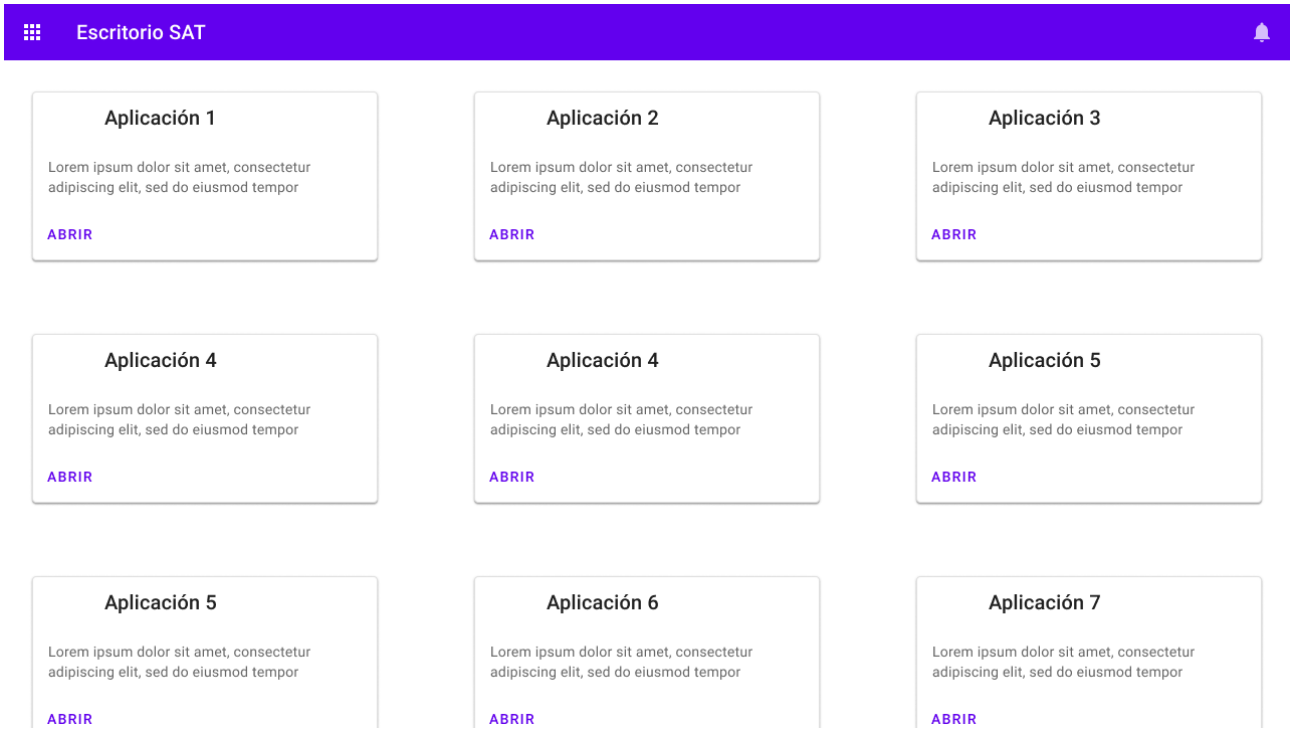

Figura 10: Prototipo de la presentación del escritorio

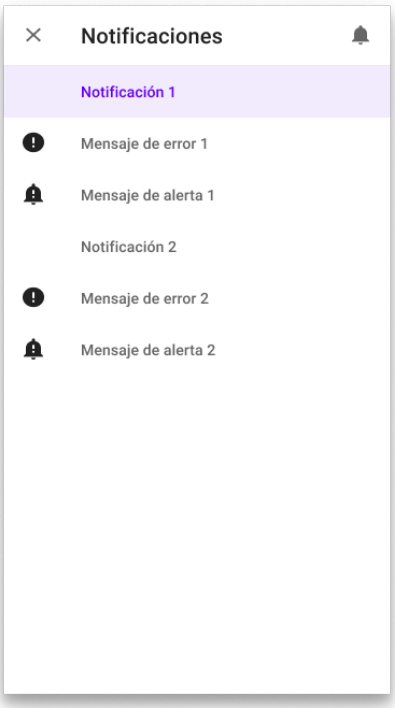

Figura 11: Prototipo del contenedor de notificaciones

12. Usabilidad

Para el aspecto de la usabilidad se han tenido en cuenta las recomendaciones de presentación que ofrecían desde el área funcional en la presentación de la necesidad de la aplicación.

La presentación de la información en esta solución, al tratarse de una aplicación de trabajo, está muy limitada en cuanto a tamaños de interfaz que se toman en consideración.

Puesto que el medio de uso de la misma va a ser a través de las pantallas en los puestos de trabajo de atención de llamadas, no se ha tenido en cuenta la presentación *responsive* para adaptar el interfaz a distintos tamaños de pantalla, desde móviles a tabletas o pantallas de ordenador.

Se ha tenido en cuenta una cierta variabilidad en las resoluciones de las pantallas, que oscilan entre un tamaño de 1280x1024 para las pantallas de 19" en formato cuadrado, hasta 1920x1080 para las pantallas panorámicas con resolución FullHD.

12.1. Recomendaciones de diseño de interfaces

Para la realización de la parte de *frontend* de la aplicación se han seguido las recomendaciones de diseño de interfaces del Nielsen-Norman Group (7).

De las 10 recomendaciones que hacen para el diseño de interfaces, las más destacadas en esta aplicación son las siguientes:

- Visibilidad del estado del sistema, puesto que las acciones de los usuarios son completadas con notificaciones si ha ocurrido algún error o si la acción se ha llevado a cabo correctamente. Además, se ofrece un área de notificaciones para otros eventos ocurridos en el sistema.
- Control del usuario y libertad, mediante un botón de cancelar en todas las acciones que provocan cambios en el estado del sistema. De este modo el usuario puede salir de la pantalla si no está seguro de querer llevar a cabo la acción.

- Consistencia y estándares, puesto que la nomenclatura utilizada para las entidades que se usan en la aplicación es la misma que la usada en el resto de aspectos del desarrollo de la tarea del operador. También, la librería usada para la interfaz de usuario, Material UI, es una librería desarrollada por Google que tiene una amplia difusión en internet. Además, esta solución sienta las bases para integrar más aplicaciones dentro de la misma plataforma que pueden aprovechar las iniciativas tomadas en ésta para la presentación de los interfaces.
- Estética y diseño minimalista, de nuevo gracias a la librería Material UI se ofrece un interfaz sencillo y simple, libre de componentes recargados o animaciones innecesarias, que agilizan el uso del interfaz.

12.2. Patrones de diseño

Como se mencionaba en la introducción de la sección, para el diseño de la interfaz de esta solución se ha tenido en cuenta una variabilidad limitada de tamaños de pantalla en los que se va a presentar la información.

Por lo tanto no se han tenido en cuenta los patrones de diseño *responsive* ya que no hay que adaptar la aplicación a tamaños de pantalla de móvil o tableta, ya que es una solución diseñada para ser usada en un entorno de trabajo integrada con otras aplicaciones.

Sin embargo, en la construcción de los interfaces sí que se han usado otros patrones de diseño como los siguientes:

- Selección de calendario (8), para seleccionar una fecha en las distintas pantallas de la aplicación.
- Realimentación en la entrada (9), para indicar al usuario de errores en la introducción de datos en los formularios antes de que se envíe la información.
- Notificaciones (10), para mostrar al usuario un apartado de la interfaz donde se listan todas las notificaciones ocurridas en el sistema.

- Modal (11), para ofrecer al usuario una ventana en la que debe realizar alguna acción antes de continuar con el flujo normal de trabajo.
- Tarjetas (12), implementadas en la pantalla principal, a través de las cuales el usuario accede a la aplicación con la que desea trabajar.
- Enlace a la página principal (13), que permite al usuario volver al escritorio para seleccionar otra aplicación con la que trabajar.
- Filtros de tabla (14), mediante los cuales el usuario puede filtrar la información presentada en la tabla de datos.
- Colores de fila alternados (15), para facilitar el seguimiento de una fila en la tabla de datos.

13. Seguridad

La seguridad del sistema ha sido un factor importante tenido en cuenta durante el diseño y desarrollo de la solución de este trabajo fin de máster.

13.1. Seguridad del entorno de ejecución

La aplicación hace uso de contenedores de Docker para su entorno de ejecución, por lo tanto la superficie expuesta al exterior es muy limitada, simplemente se expone el puerto de entrada a la aplicación.

Además, al contar con el programa desplegado dentro de un contenedor, no hay archivos o librerías que se puedan sobrescribir en el sistema de archivos, o infectar con virus.

Sin embargo, hay que prestar especial atención al origen de los contenedores, primando las distribuciones oficiales de los mismos frente a contenedores construidos por terceras personas.

Uno de los riesgos que se han detectado (16) con los contenedores es la inclusión de software malicioso en contenedores de dudoso origen, por ello es importante ceñirse a las distribuciones oficiales.

Otra opción es generar el contenedor de forma manual a partir de un archivo `Dockerfile`, sin embargo es una tarea más manual y muy propensa a fallos, y debe ser realizada con cuidado por personas con experiencia en la configuración de administración de sistemas informáticos.

13.2. Seguridad de los secretos

La única información privilegiada que se encuentra en la solución es la cadena de conexión a la base de datos que se proporciona al contenedor de servicios de *backend*.

Esta información se debe proporcionar al contenedor a través de una variable de entorno configurada en el archivo `docker-compose-build.yml`, pero hay que tener especial atención en no volcar esta información a un sistema de

control de versiones de código, ya que estaría visible para todos los usuarios de dicho sistema de control de versiones.

Este dato debe ser conocido por el personal de operación del servicio y ser configurado en el archivo en el momento de despliegue al entorno de producción.

13.3. Seguridad en la integridad de la aplicación

Para el desarrollo de los servicios de *backend* se ha usado la librería RepoDB para la interacción con la base de datos.

Dicha librería es un *Object Relational Mapper* (ORM) que ofrece construcciones estandarizadas para acceder a los datos de la base de datos, lo cual evita todo un campo de vulnerabilidades como el de la inyección de SQL (17).

Además, la aplicación ofrece actualmente dos niveles de acceso con funcionalidad diferenciada para cada uno de ellos. Las acciones a realizar se validan tanto en el área de *frontend* como en el área de *backend* para evitar que se puedan elaborar peticiones a mano y manipular de forma no autorizada los datos de la aplicación.

Para la validación de dichas acciones se cuenta con un token de seguridad el cual, además de los datos del usuario incorpora una firma que autentica que los datos presentes en el token son ciertos.

14. Requisitos de instalación

Este proyecto se ha desarrollado íntegramente usando contenedores de Docker, por lo que la instalación de los requisitos para hacerlo funcionar es muy reducida.

Para la instalación de este proyecto hacen falta los siguientes componentes:

- Motor de contenedores Docker
- Orquestador de contenedores Docker Compose
- Sistema operativo Windows, MacOS o Linux

Puesto Docker esta disponible para una amplia variedad de arquitecturas, se pueden generar los contenedores tanto en arquitecturas X86-64 como ARM, siendo estas las más habituales en equipos de escritorio y servidores.

Para el desarrollo de este proyecto se ha usado un contenedor de Microsoft SQL Server 2019 para hacer de motor de base de datos para los servicios de backend. En el entorno de despliegue en producción se cuenta con un motor de base de datos que contiene la información necesaria de todo el sistema en el que se va a integrar este proyecto, por lo que en este caso no se requiere disponer de dicho motor de base de datos, ya que actualmente esta presente.

Sin embargo, si se fuera a instalar este proyecto en otro entorno distinto del entorno de producción objetivo, sí sería necesario contar con, además del motor de base de datos mencionado anteriormente, el esquema de datos del resto de entidades con las que se relaciona este proyecto y que son ajenas al mismo.

15. Instrucciones de instalación

Para instalar el software hay que llevar a cabo una serie de pasos que se explican a continuación.

15.1. Configuración de la aplicación

Para la configuración de la aplicación es necesario indicar la cadena de conexión de la base de datos para los servicios de *backend*.

Esta cadena se encuentra definida como una variable de entorno en el archivo `docker-compose-build.yml` y usa actualmente el motor de base de datos incorporado al entorno de ejecución, pero si se cuenta con un motor de base de datos Microsoft SQL Server con el esquema necesario para esta aplicación, simplemente hay que sustituir los parámetros de conexión en la cadena de conexión.

15.2. Generación de los contenedores

Como se ha mencionado en el apartado anterior, para el desarrollo de este proyecto se han usado contenedores de Docker, e igualmente para el entorno de producción se ha configurado un archivo `docker-compose-build.yml` para generar los contenedores de este entorno.

Por lo tanto, para generar los contenedores hay que ejecutar el siguiente comando:

```
docker-compose -f docker-compose-build.yml build
```

Cuando este comando haya terminado de ejecutar se habrán generado los contenedores de cada uno de los componentes, que son los siguientes:

- Contenedor *backend* que contiene los servicios desarrollados en .NET Core. Estos servicios se sirven a través del puerto 5000 del contenedor, pero el acceso no será directo al mismo.
- Contenedor *frontend* que contiene un servidor Apache Httpd, el cual

esta configurado para exponer el puerto 80 la aplicación y permitir el acceso a la aplicación Angular a partir de la ruta `/escritorio-sat` o los servicios de *backend* a partir de la ruta `/servicios`.

15.3. Ejecución de la aplicación

Para ejecutar la aplicación en los contenedores de Docker que se han preparado en el paso anterior hay que ejecutar el siguiente comando:

```
docker-compose -f docker-compose-build.yml -d up
```

Este comando iniciará todos los contenedores que forman parte de la aplicación, y el acceso a la misma será a través del puerto 80 de la máquina en la que se esta ejecutando.

Al indicar el parámetro `-d`, los contenedores se iniciarán en segundo plano y la consola no se quedará bloqueada con la ejecución de los mismos.

Habrá que tener en cuenta la configuración de cortafuegos para permitir la entrada de las peticiones, pero esa configuración queda fuera del ámbito del proyecto ya que la aplicación se puede ejecutar en una amplia variedad de entornos y no se puede dar una configuración que cubra todas las posibilidades.

15.4. Mantenimiento de la aplicación

Los contenedores están configurados para que se reinicien automáticamente si hay cualquier problema, por lo que la ejecución de la aplicación estaría asegurada incluso en el caso de existir errores graves.

En cualquier caso, las aplicaciones vuelcan los registros de ejecución a través de la consola de salida del contenedor, y estos registros se pueden visualizar con el siguiente comando de Docker Compose:

```
docker-compose -f docker-compose-build.yml logs <contenedor>
```

Donde el parámetro `<contenedor>` hay que sustituirlo por el nombre del contenedor del cual se quieran ver los registros, a saber:

- *frontend*, para ver los registros del servidor Apache HTTP.
- *backend*, para ver los registros de la aplicación .NET Core.
- *sqlserver*, para ver los registros del motor de base de datos Microsoft SQL Server, si se ha usado este contenedor.

15.5. Actualización de la aplicación

En el caso en que se realicen modificaciones en alguna de las aplicaciones que componen la solución, simplemente con volver a generar los contenedores en el primer paso sería suficiente para actualizar los servicios, y a continuación reiniciar de nuevo los contenedores en ejecución y borrar las imágenes antiguas.

Estas operaciones se pueden combinar con los siguientes comandos:

```
docker-compose -f docker-compose-build.yml --force-recreate --build -d up
docker image prune -f
```

En el primer comando se inician los contenedores de nuevo forzando una reconstrucción de los mismos, y con el segundo comando se eliminan las imágenes de los contenedores de la versión antigua.

16. Instrucciones de uso

Para usar la aplicación hay que cargar la dirección correspondiente en el navegador web.

Una vez cargada la aplicación, se presenta al usuario con una pantalla en la que hay una lista de módulos (en este trabajo de fin de máster se ha desarrollado uno para la gestión de las llamadas estacionadas, pero se ha construido la aplicación de modo que en un futuro se puedan incorporar nuevos módulos) en el espacio central y un botón con el icono de una campana para mostrar u ocultar las notificaciones del sistema.

Cuando el usuario pulsa sobre el botón “Abrir” del módulo de gestión de llamadas técnicas, se le presenta una tabla en el espacio central con los datos sobre llamadas estacionadas actualmente existentes en el sistema.

En el lado derecho se muestra un apartado con los detalles del registro seleccionado en la tabla y una serie de botones para llevar a cabo acciones según su nivel de acceso.

Para los usuarios con perfil operador la única acción que se puede llevar a cabo es la realización de la llamada estacionada, que consiste en la selección de un motivo de realización para guardar el dato en el servidor y eliminar el registro de la tabla.

Para los usuarios con perfil supervisor se ofrecen las siguientes funcionalidades:

- Nuevo registro, para crear un nuevo registro de llamada estacionada.
- Editar registro, para modificar un registro de llamada estacionada.
- Eliminar registro, para eliminar un registro de llamada estacionada.
- Asignar registro, para asignar un registro de llamada estacionada a un operador.
- Desasignar registro, para cancelar la asignación de un registro a un operador.

- Realizar registro, para, al igual que el operador, poder insertar el histórico de realización de llamada estacionada.

Además, cuando un registro esta asignado a un operador y se aproxima la hora de la llamada, se muestra una notificación en el componente de notificaciones mencionado anteriormente, a modo de aviso.

17. Proyección a futuro

El proyecto desarrollado en este trabajo de fin de máster será ampliado a futuro puesto que una de las premisas del mismo era ofrecer un entorno dentro del cual se puedan desarrollar módulos adicionales.

Sin embargo, antes de acometer esta ampliación de funcionalidades, será necesario completar las tareas que han quedado pendientes por falta de tiempo de desarrollo.

Este Escritorio SAT permitirá desarrollar nuevos módulos que implementarán nuevas funcionalidades dentro de las necesidades funcionales, no solamente de aspectos relacionados con la gestión de las llamadas o sus datos asociados, sino también para ofrecer aplicaciones web para gestionar datos administrativos para los departamentos que se encargan de tareas como la gestión de personal, gestión de la formación y demás departamentos.

Por otro lado, el módulo desarrollado actualmente para la gestión de las llamadas estacionadas se encuentra ya en proceso de revisión para añadir nuevas funcionalidades en un futuro cercano.

18. Conclusiones

El desarrollo de este proyecto ha permitido cubrir una necesidad en la organización de esta tarea del servicio de atención telefónica, por lo tanto se considera un éxito en este aspecto.

La tarea era gestionada actualmente mediante archivos de Microsoft Excel y hojas impresas, y al disponer de una aplicación web que integra toda la información necesaria se ha optimizado ampliamente el desarrollo de la misma.

La existencia de esta aplicación permite el ahorro de entre cinco a ocho horas diarias de trabajo en la gestión de las llamadas estacionadas por parte del personal teleoperador y, sobretodo, del personal supervisor, que tiene una mayor carga de trabajo en la organización de esta tarea.

Además, al contar con los datos de ambas centrales de atención telefónica, la coordinación entre ambas ha mejorado al no necesitar una llamada telefónica para realizar el seguimiento de la tarea de la otra central, puesto que la propia aplicación puede mostrar los datos de ambas centrales.

Además, al haber desarrollado la plataforma web del Escritorio SAT, se dispone de un entorno sobre el cual se pueden desarrollar más aplicaciones web que permitan la automatización o estructuración de otras tareas.

Actualmente se encuentran en estudio otras tareas que en un futuro serán incorporadas a esta plataforma.

En el aspecto técnico, se ha desarrollado una aplicación siguiendo las nuevas tendencias de desarrollo y despliegue usando contenedores.

El uso de contenedores para el desarrollo facilita y agiliza la tarea del desarrollador al ofrecer un entorno encapsulado para creación de la solución, ya que facilita la coexistencia de varios entornos de desarrollo simultáneos sin causar conflictos entre distintas versiones de las librerías.

Y el uso de contenedores para el despliegue de las aplicaciones facilita igualmente la puesta en producción ya que engloba en una imagen la solución

completa sin dependencias externas, por lo que puede ser puesta en marcha en un entorno de producción en un corto espacio de tiempo.

Anexo 1. Entregables del proyecto

Informe_Autoevaluacion_TFM_Estudiante_es.doc

PAC_FINAL_mem_SerranoThode_Damian.pdf

PAC_FINAL_prj_SerranoThode_Damian.zip

PAC_FINAL_prs_SerranoThode_Damian.pdf

Dentro del archivo comprimido del código fuente del proyecto:

Carpeta /frontend: El código fuente de la aplicación de frontend.

Carpeta /backend: El código fuente de la aplicación de backend.

Carpeta /sqlserver: El script de configuración de la base de datos y el archivo Dockerfile para el contenedor.

Carpeta /websockets: El archivo Dockerfile para el contenedor de ActiveMQ.

Carpeta /httpd: El archivo Dockerfile y la configuración para el servidor Apache HTTPD del entorno de desarrollo.

Archivo docker-compose-dev.yml: El archivo de orquestación de contenedores para el entorno de desarrollo.

Archivo docker-compose-build.yml: El archivo de orquestación de contenedores para el entorno de producción.

Anexo 2. Definición de funcionalidades

En este capítulo se van a definir las funcionalidades que se desean implementar en las aplicaciones mediante historias de usuario. Estas historias han sido definidas a partir de una serie de reuniones en las que se han extraído las necesidades y se han plasmado estos requisitos en historias de usuario.

Requisitos del escritorio de aplicaciones

Los requisitos del escritorio de aplicaciones se pueden diferenciar según la entidad que interactúe con la aplicación. Por lo tanto, se van a clasificar en las secciones presentadas a continuación.

Usuario de la aplicación

- (HUEWU01) Como usuario, quiero ver todas las aplicaciones a las que tenga permiso de acceso.
- (HUEWU02) Como usuario, quiero que las aplicaciones a las que no tengo permiso de acceso no aparezcan en el escritorio.
- (HUEWU03) Como usuario, quiero poder abrir una aplicación.
- (HUEWU04) Como usuario, quiero ver las notificaciones de las aplicaciones.
- (HUEWU05) Como usuario, cuando esté usando una aplicación, quiero volver al escritorio para abrir otra aplicación distinta.

Componente de la aplicación escritorio

- (HUEWC01) Como un componente, quiero registrar la aplicación en el catálogo de aplicaciones disponibles para que un usuario la pueda abrir.
- (HUEWC02) Como un componente, quiero que solamente un usuario con los permisos adecuados pueda abrir la aplicación.
- (HUEWC03) Como un componente, quiero poder mostrar el contenido en el espacio disponible para aplicaciones.

Requisitos de la aplicación de llamadas estacionadas

La aplicación de llamadas estacionadas, al igual que el escritorio de aplicaciones, también puede clasificarse según la entidad que interactúa con el sistema, por lo tanto se presentan igualmente las siguientes secciones a continuación.

Operador telefónico

- (HULLOP01) Como operador, quiero ver las llamadas estacionadas que tengo asignadas.
- (HULLOP02) Como operador, quiero ver las llamadas asignadas a otros operadores en otro color.
- (HULLOP03) Como operador, quiero poder abrir los detalles de la vivienda que corresponde con un registro del listado.
- (HULLOP04) Como operador, quiero poder llamar al usuario correspondiente a un registro del listado.
- (HULLOP05) Como operador, quiero que me notifique cuando se aproxime la hora de la siguiente llamada que tengo asignada.
- (HULLOP06) Como operador, quiero sugerir la inserción de un nuevo registro de llamada estacionada.
- (HULLOP07) Como operador, quiero sugerir modificaciones a un registro de llamada estacionada.
- (HULLOP08) Como operador, quiero sugerir la eliminación de un registro de llamada estacionada.
- (HULLOP09) Como operador, quiero que una llamada estacionada realizada se elimine del listado.
- (HULLOP10) Como operador, quiero dar por realizado una llamada estacionada aunque no haya podido contactar con el usuario.

Supervisor

- (HULLS01) Como supervisor, quiero añadir registros en el listado de llamadas estacionadas.

- (HULLS02) Como supervisor, quiero ver la lista de llamadas estacionadas de mi central.
- (HULLS03) Como supervisor, quiero ver la lista de llamadas estacionadas de otras centrales.
- (HULLS04) Como supervisor, quiero modificar un registro del listado de llamadas estacionadas.
- (HULLS05) Como supervisor, quiero eliminar un registro del listado de llamadas estacionadas.
- (HULLS06) Como supervisor, quiero ver la carga de trabajo de llamadas estacionadas a realizar.
- (HULLS07) Como supervisor, quiero repartir las llamadas a un grupo de operadores.
- (HULLS08) Como supervisor, quiero eliminar las asignaciones de llamadas estacionadas.
- (HULLS09) Como supervisor, quiero filtrar la lista de llamadas estacionadas.
- (HULLS10) Como supervisor, quiero obtener informes de las llamadas estacionadas realizadas en el turno.
- (HULLS11) Como supervisor, quiero revisar las sugerencias de nuevos registros, registros modificados o eliminaciones.
- (HULLS12) Como supervisor, quiero exportar el listado de llamadas que estoy visualizando a un archivo.
- (HULLS13) Como supervisor, quiero configurar las tipologías de las llamadas estacionadas.

Componente de la aplicación

- (HULLC01) Como componente, quiero recibir notificaciones del servidor para actualizar los registros del listado de llamadas.
- (HULLC02) Como componente, quiero dejar registro de las acciones realizadas en la aplicación.

Anexo 3. Bibliografía

1. Agile software development. *Wikipedia* [en línea]. 2021. [Consulta: 7 junio 2021]. Disponible en: https://en.wikipedia.org/w/index.php?title=Agile_software_development&oldid=1026872969Page Version ID: 1026872969

2. The C4 model for visualising software architecture. [en línea]. [Consulta: 4 junio 2021]. Disponible en: https://c4model.com/

3. BOOCH, Grady, RUMBAUGH, James y JACOBSON, Ivar. *The unified modeling language user guide*. 2nd ed. Upper Saddle River, NJ : Addison-Wesley, 2005. ISBN 9780321267979.

4. KRUCHTEN, P. B. The 4+1 View Model of architecture. *IEEE Software*. noviembre 1995. Vol. 12, no. 6, p. 42-50. DOI 10.1109/52.469759.

5. RepoDB. RepoDB. [en línea]. [Consulta: 4 junio 2021]. Disponible en: https://repodb.net/

6. STOMP. [en línea]. [Consulta: 7 junio 2021]. Disponible en: https://stomp.github.io/

7. EXPERIENCE, World Leaders in Research-Based User. 10 usability heuristics for user interface design. Nielsen norman group. [en línea]. [Consulta: 7 junio 2021]. Disponible en: https://www.nngroup.com/articles/ten-usability-heuristics/

8. Calendar Picker design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/CalendarPicker

9. Input Feedback design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/InputFeedback

10. Notifications design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/notifications

11. Modal design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/modal-windows

12. Cards design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en:

http://ui-patterns.com/patterns/cards

13. Home Link design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/HomeLink

14. Table Filter design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/TableFilter

15. Alternating Row Colors design pattern. [en línea]. [Consulta: 7 junio 2021]. Disponible en: http://ui-patterns.com/patterns/AlternatingRowColors

16. Half of 4 million public docker hub images found to have critical vulnerabilities. InfoQ. [en línea]. [Consulta: 7 junio 2021]. Disponible en: https://www.infoq.com/news/2020/12/dockerhub-image-vulnerabilities/

17. SQL injection OWASP. [en línea]. [Consulta: 7 junio 2021]. Disponible en: https://owasp.org/www-community/attacks/SQL_Injection