Trabajo Fin de Grado

Ingeniería de las Tecnologías de Telecomunicación

Mención en Telemática

Diseño y desarrollo de un bot Telegram para la gestión de actividades asistenciales

Autor: Luis Marín Peña

Tutora: Isabel Román Martínez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Grado
Ingeniería de las Tecnologías de Telecomunicación
Mención en Telemática

Diseño y desarrollo de un bot Telegram para la gestión de actividades asistenciales

Autor:
Luis Marín Peña

Tutora:
Isabel Román Martínez

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Grado: Diseño y desarrollo de un bot Telegram para la gestión de actividades asistenciales

Autor: Luis Marín Peña

Tutora: Isabel Román Martínez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El secretario del Tribunal

AGRADECIMIENTOS

Este trabajo ha sido posible gracias a la inestimable ayuda de las personas que me han apoyado para terminarlo.

Aprovecho este espacio para agradecer a mi tutora, Isabel Román, por su paciencia y dedicación que ha tenido conmigo, por la estructuración que me ha ayudado a tener, y, sobre todo, por saber guiarme a lo largo de la realización de este proyecto.

Quisiera agradecer a Beatriz por estar animándome y haciéndome más fácil que trabajase en este proyecto, aun cuando se dificultaba por la falta de tiempo.

También quiero agradecer a mi amigo Dorian por invertir su tiempo en hacer también su trabajo conmigo para animarme a avanzarlo con más motivación.

Agradezco a mi familia por haberme dado la posibilidad de llevar los estudios hasta el final, con su apoyo moral, emocional y económico.

A mis amigos y compañeros de clase por haberme acompañado en este viaje de tantos años.

A los profesores de la Escuela y de bachiller que me dieron entusiasmo e ilusión por la ingeniería, que permitió que llegase hasta aquí.

Sin todos ellos, este trabajo no sería posible.

RESUMEN

El objetivo de este proyecto es diseñar e implementar una herramienta que permita facilitar la comunicación entre el personal sanitario a la hora de coordinar los cambios en las actividades asistenciales.

Actualmente se está desarrollando un sistema que permite la gestión de actividades asistenciales [1] [2] para hacer más eficiente la planificación de éstas en el Servicio de Medicina Interna del Hospital Universitario Virgen Macarena (HUVM)

Este sistema proporciona al gestor de dicha unidad una aplicación para planificar las actividades asistenciales de su servicio.

Dicho sistema tiene 3 componentes principales:

- Un Servicio REST que ofrece acceso a la información del sistema
- Una aplicación web que ofrece una interfaz de acceso al gestor
- Servicio de planificación de actividades asistenciales acorde a restricciones

El objetivo de este proyecto es ampliar el sistema anterior, diseñando e implementando una herramienta que permita facilitar la comunicación entre el personal sanitario a la hora de coordinar los cambios en las actividades asistenciales, y que sea accesible desde un teléfono móvil. Ésta debe facilitar a los usuarios la consulta y modificación de las actividades asistenciales en las que participan.

Dicha herramienta, va a interactuar con el servicio REST que está actualmente en desarrollo, que será el que cree los calendarios de las planificaciones mensuales. Estos calendarios van a ser utilizados por la herramienta que en este trabajo se va a desarrollar.

También interactuará con los calendarios, conectándose a un servicio CalDAV [3] donde se alojan los mismos, para gestionar los asistentes a las actividades

Para ello se va a diseñar, programar y desplegar un servicio automatizado de gestión, utilizando un bot de Telegram que se encargará de comunicar al miembro del servicio con el servicio REST de gestión anteriormente mencionado.

Se pretende que el sistema diseñado sea suficientemente genérico como para poderse aplicar en otras instituciones y organizaciones.

Para este desarrollo se va a utilizar el lenguaje de programación Python [4], por ofrecer una sencilla portabilidad de código. Esto permite que se pueda alojar en cualquier sistema operativo.

También utilizaremos una API desarrollada por el equipo de Telegram [5] que nos facilitará la comunicación con los usuarios del servicio mediante cualquier dispositivo que disponga de la aplicación Telegram o de un navegador web.

SUMMARY

Currently, a system is being developed that allows the management of doctor's shifts [1] [2] to make the planning of shifts in health services more efficient, which is aimed at the Internal Medicine Department of the Hospital Universitario Virgen Macarena (HUVM).

This system provides the manager of this unit with an application to plan the doctor's shifts of his service.

This system has 3 main components:

- A REST Service that provides access to the system information.
- A web application that provides an access interface to the scheduler.
- A shift planning service according to the scheduler requirements.

The objective of this project is to extend the previous system by providing a communication tool accessible from a cell phone. It should be able to modify and manage the shifts of the service members, make changes to these shifts or access to the planning of their work shifts.

This tool will interact with the REST service that is currently under development, which will create the monthly planning calendars. These calendars will be used by the tool that will be developed in this work.

It will also interact with the calendars, connecting to a CalDAV [3] service where the calendars are hosted, to manage the attendees to the shifts.

For this purpose, an automated management service is going to be designed, programmed, and deployed, using a Telegram bot that will be in charge of communicating the service member with the aforementioned management REST service.

The designed system is intended to be sufficiently generic to be applied in other institutions and organizations.

The Python programming language [4] is going to be used for this development, since it offers a simple code portability. This allows it to be hosted on any operating system.

We will also use an API developed by the Telegram team [5] that will facilitate communication with users of the service through any device that has the Telegram application or a web browser.

ÍNDICE

AGRADECIMIENTOS **7**
RESUMEN **9**
SUMMARY **10**
ÍNDICE **12**
ÍNDICE DE TABLAS **13**
ÍNDICE DE CÓDIGO **13**
ÍNDICE DE ILUSTRACIONES **14**
GLOSARIO **16**
1 INTRODUCCIÓN **18**
1.1 PARTICIPANTES IDENTIFICADOS 18
1.2 DESCRIPCIÓN DEL PROCESO ACTUAL 18
1.3 REGLAS DE NEGOCIO IDENTIFICADAS 19
1.4 ALCANCE DEL PROYECTO 19
2 TECNOLOGÍAS PRINCIPALES DEL PROYECTO **20**
2.1 UML 20
2.2 PYTHON 20
2.3 GIT 21
2.4 REST 22
2.5 API PARA BOTS DE TELEGRAM 23
2.6 DOCKER 24
2.7 ICALENDAR 24
2.7.1 CalDAV *24*
2.7.2 DAViCal *24*
3 REQUISITOS **25**
3.1 ACTORES 25
3.2 CASO DE USO 26
3.2.1 Situación actual *26*
3.2.2 Situación deseada *28*
3.3 REQUISITOS FUNCIONALES 31
3.4 REQUISITOS NO FUNCIONALES 31
4 TRABAJO REALIZADO **32**
4.1 ESTRUCTURA 32
4.2 COMPORTAMIENTO 36
4.2.1 Casos de uso *36*
4.3 PRUEBAS DE UNIDAD 47
4.4 COMENTARIOS DE CÓDIGO 48
4.5 DISTRIBUCIÓN DE LA APLICACIÓN 49
4.5.1 Dockerhub *49*
4.6 DESPLIEGUE DE LA APLICACIÓN 50
5 LÍNEAS DE CONTINUACIÓN Y CONCLUSIONES **51**
5.1 AGREGAR FUNCIONALIDAD PARA RESERVAR CITAS 51
5.2 AÑADIR ENVÍOS DE CAMBIO DEL CALENDARIO A LA API REST 51
5.3 SIMPLIFICACIÓN DE CONFIGURACIÓN 51
5.4 CONCLUSIONES 51
6 ANEXOS **52**
ANEXO A - GUÍA DE ACTIVACIÓN DE UNA MÁQUINA VIRTUAL EN AZURE 52
Sistemas Operativos disponibles *56*
Características que tener en cuenta *57*
ANEXO B - DOCUMENTACIÓN DE CÓDIGO A PARTIR DE COMENTARIOS 59
7 REFERENCIAS **61**

ÍNDICE DE TABLAS

Tabla 1: A-01 - Ofertante 25
Tabla 2: A-02 – Demandante 25
Tabla 3: A-03 – Gestor 25
Tabla 4: A-04 - Participante 25
Tabla 5: RF-01 - Identificación de usuario 31
Tabla 6: RF-02 - Iniciación de la aplicación 31
Tabla 7: RNF-01 - Comentarios de código 31
Tabla 8: RNF-02 - Pruebas de unidad 31
Tabla 9: RNF-03 - Trazas de código 31

ÍNDICE DE CÓDIGO

Código 1: Código ejemplo test 47
Código 2: Función de ejemplo comentada con la guía de estilo de Google 48
Código 3: Dockerfile para distribución de aplicación 49
Anexo B - Código 4: Fichero mkdocs.yml 60

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Relación botón contextual, mensaje, canal y chat 23
Ilustración 2: Diagrama actividad de intercambio de actividad, situación actual 26
Ilustración 3: Estados de la actividad, situación actual 27
Ilustración 4: Diagrama de casos de uso, situación deseada 28
Ilustración 5: Diagrama actividad de cesión de actividad, situación deseada 29
Ilustración 6: Estados de la actividad, situación deseada 30
Ilustración 7: Diagrama de clases de la aplicación 33
Ilustración 8: Secuencia de identificación en bot, situación deseada 36
Ilustración 9: Interfaz de la aplicación en Telegram 1 37
Ilustración 10: Teclado de respuestas 37
Ilustración 11:Teclado de respuestas de administrador 37
Ilustración 12:Interfaz de Telegram, Participante consulta actividades propias 38
Ilustración 13: Interfaz de Telegram, ofertante cede una actividad con éxito 39
Ilustración 14: Publicación de Actividad cedida en el canal de Telegram 40
Ilustración 15: Actividad en la que se ha inscrito el demandante 41
Ilustración 16: Publicación de la propuesta de cesión en canal de administradores 41
Ilustración 17: Diagrama de secuencia de cesión de actividad, situación deseada 42
Ilustración 18: Diagrama de secuencia de intercambio de actividad, situación deseada 43
Ilustración 19: Cancelación de una oferta 45
Ilustración 20: Cancelación de demanda 46
Ilustración 21: Resultado de test con éxito y con error 47
Ilustración 22: Componentes del servicio 50
Anexo A - Ilustración 23: Azure For Education, inscripción al servicio 52
Anexo A - Ilustración 24: Azure for Education, verificación de identidad 53
Anexo A - Ilustración 25: Azure for Education: Portal principal 53
Anexo A - Ilustración 26: Activación del perfil de Azure For Education 54
Anexo A - Ilustración 27: Azure Free Services 54
Anexo A - Ilustración 28: Azure Free Services, detalle de los servicios 55
Anexo A - Ilustración 29: Creacion de una Máquina Virtual 56
Anexo A - Ilustración 30: Datos de una Máquina Virtual 57
Anexo A - Ilustración 31: Máquina virtual ya creada 58
Anexo A - Ilustración 32: Configuración de red de Máquina Virtual 58
Anexo B - Ilustración 33: Estructura del directorio del repositorio con MkDocs ejecutado 59

GLOSARIO

A

API

Interfaz de Programación de Aplicaciones. Esta interfaz permite el acceso a datos y/o funcionalidades de otros servicios mediante uso de funcionalidad documentada para que otras aplicaciones puedan comunicarse con el servicio. 22

Asistente

Representa un participante en una actividad asistencial. 24

B

backports

Hacer modificaciones a una versión antigua para añadir funcionalidades o parches de versiones más nuevas 56

Botones contextuales de mensaje

Botones situados debajo de un mensaje enviado por el bot. Permiten acciones contextuales relacionadas con el mensaje. 35

Botones de comandos de Menú

Botones necesarios para poder iniciar la aplicación la primera vez que se utiliza. 35

Botones de teclado de respuestas.

Botones utilizados para que los usuarios puedan enviar comandos al bot, sin esperar una acción previa del bot 35

D

demandante

Trabajador que desea participar en una actividad ofrecida por otro compañero de su mismo servicio 18

E

Evento

Representa una actividad asistencial en el calendario. Se expresa en formato iCal. Se compone de Resumen o Nombre del evento, descripción del evento, fecha de inicio de evento, fecha de fin de evento y asistentes. 34

G

gestor

Persona que se encarga de planificar las actividades de un servicio sanitario, y de autorizar los intercambios de las actividades entre empleados 18

H

HPC

High Powered Computing o Computación de Alto Rendimiento, es la práctica de unir recursos computacionales para su uso en aplicaciones o servicios que realizan operaciones. [22] 56

HUVM

Hospital Universitario Virgen Macarena 9

L

LTS

Long Term Support o Soporte a Largo Plazo. Es una nomenclatura para versiones con soporte mayor al habitual. En caso de Ubuntu, son 5 años frente a los 9 meses habituales 56

O

ofertante

Asistente de una actividad que desea ceder o intercambiar su puesto en la misma con un compañero del servicio. Para ello oferta su puesto para cesión o intercambio 18

S

SRIOV

La virtualización de E/S de raíz única (SR-IOV) es una extensión de la especificación PCI Express (PCIe). SR-IOV permite que un dispositivo de interconexión física, como un adaptador de red de varios puertos, separe el acceso a sus recursos y aparezca como varios dispositivos físicos distintos ante el hipervisor o el sistema operativo invitado [23]. 56

1 INTRODUCCIÓN

El objetivo de este proyecto es diseñar e implementar una herramienta que permita facilitar la comunicación entre el personal sanitario a la hora de coordinar los cambios en las actividades asistenciales.

Dado que este objetivo es lo suficientemente general, se puede aplicar este proyecto a otros servicios con los mismos requisitos.

1.1 Participantes identificados

Para aclarar la situación actual, vamos a definir tres figuras o perfiles que nos ayudarán a describirla:

1. Llamaremos al usuario que propone un cambio de actividad "ofertante"
2. Llamaremos al usuario que desea la actividad que un compañero ofrece "demandante"
3. Llamaremos "gestor" al gestor del servicio sanitario, entre sus funciones está la planificación de las actividades del servicio y autorizar los cambios acordados entre el demandante y ofertante

1.2 Descripción del proceso actual

Se describe a continuación el procedimiento que se sigue actualmente para el intercambio de actividades.

Los trabajadores conversan entre sí, ya sea en grupo o individualmente, proponiendo cambiar sus actividades a la espera de que otra persona interesada en ésta ofrezca una propia a cambio. Habitualmente se utilizan aplicaciones de mensajería como WhatsApp para este menester. Una vez ambas partes están de acuerdo, se propone el intercambio al gestor del servicio, se firma una autorización de cambio si este es aceptado, y se completa el intercambio.

El procedimiento actual ofrece la flexibilidad necesaria para que cualquier trabajador pueda cambiar sus actividades con otro, con la única restricción de no dejar una actividad con menos participantes que tuviera originalmente.

Sin embargo, su funcionamiento provoca que dicho intercambio sea lento e ineficiente, principalmente por los siguientes motivos:

- Depende de cuánto tiempo el demandante tarde en conocer dicha propuesta de cambio.
- Una vez se llega a un acuerdo, hacerle llegar la propuesta al gestor requiere normalmente acudir a su oficina.

1.3 Reglas de negocio identificadas

- Un cambio de actividad debe ser aprobado por el *gestor* del servicio
- Las actividades afectadas en un cambio deben tener la misma cantidad de participantes que tuvieran originalmente para que el cambio sea aprobado.
- Un *ofertante* puede cancelar la oferta mientras el gestor no acepte o deniegue el cambio de actividad
- Un *demandante* puede cancelar su demanda mientras el gestor no acepte o deniegue el cambio de actividad

1.4 Alcance del proyecto

El objeto de este proyecto es automatizar el procedimiento de cambio de actividad lo máximo posible, de manera que se pueda coordinar la acción de los tres participantes.

Dado que la actividad que más habitualmente se intercambia o cede es la realización de guardias, en la memoria se hace a veces referencia a esta actividad en concreto. Aunque los procedimientos se podrán aplicar a cualquier otro tipo de actividad asistencial. Se distinguen dos procedimientos principales: la cesión de una actividad y el intercambio de actividades.

El usuario (*ofertante)* podrá activar una propuesta de cesión en una de sus actividades. Dicha actividad podrá ser escogida por otros usuarios (*demandantes*) para proponer una nueva asignación de dicha actividad. Cuando se active una propuesta de cesión o de intercambio, aparecerá dicha oferta en un canal de Telegram para que pueda ser demandada.

Una vez se cubran los cupos de una actividad en propuesta de cesión, pasará a ser actividad en propuesta de cambio pendiente de aceptar o denegar, a la espera de que el gestor tome una decisión sobre ese cambio.

Otra función que se pretende cubrir es el intercambio de actividades. El ofertante podrá ofrecer una actividad, y cuando un demandante la solicite para cambiarla, deberá ofrecer actividades a cambio. El ofertante escogería una de las actividades del demandante, y se enviaría la propuesta completa al gestor para que la acepte o deniegue.

El *gestor* podrá recibir notificaciones en un canal de Telegram de propuestas de cambio y tendrá la posibilidad de aceptar o denegar dicho cambio de actividad desde la propia aplicación de Telegram.

La herramienta también servirá al usuario para obtener las actividades asistenciales que tiene actualmente asignadas, y debe permitir retractar las propuestas antes de que el gestor las acepte o deniegue.

Una vez el cambio esté aceptado por el gestor, se hará efectivo en el calendario de actividades del mes en curso.

2 TECNOLOGÍAS PRINCIPALES DEL PROYECTO

En este capítulo vamos a dar una visión general sobre las tecnologías utilizadas durante la planificación, diseño, desarrollo e implementación del proyecto, necesarias para la comprensión del mismo. De manera resumida, también se aportarán algunas explicaciones y ejemplos de estas.

2.1 UML

El Lenguaje de Modelado Unificado (Unified Modelling Language en inglés) es un lenguaje de modelado utilizado en ingeniería de software, que permite una forma estructurada y universal de representar el diseño de un software, incluso antes de su programación.

Con UML vamos a representar, mediante diagramas, aspectos del software que se pretenden destacar para detallar su funcionamiento interno.

2.2 Python

Python es un lenguaje de programación, que permite su uso tanto como lenguaje orientado a objetos como lenguaje procedimental.

Sus principales características son:

- Es un lenguaje interpretado, y existen intérpretes para Windows, Linux, Mac, iOS y Android, entre otros.
- Está tipado dinámicamente, lo que significa que una variable puede cambiar su tipo en tiempo de ejecución.
- Sus bibliotecas estándar (en Python las bibliotecas se conocen como *módulos*) incluyen funciones para cubrir muchos casos de uso, desde manejo de bases de datos relacionales, generación de números aleatorios, funciones matemáticas, accesos a funciones del sistema operativo o pruebas de unidad, entre otras.
- Cuenta con el Python Package Index (índice de paquetes de Python o su acrónimo *PyPi*). Son módulos creados por la comunidad de desarrolladores de Python, y añaden funcionalidades no incluidas en las bibliotecas estándar de Python. Este índice cuenta con cerca de 400.000 proyectos de módulos en línea.

En este proyecto vamos a hacer uso de los siguientes módulos del PyPi:

- *python-telegram-bot* [6]: Ofrece clases y funciones para comunicar el programa desarrollado con las API de Telegram.
- *pyYAML* [7]: Ofrece funciones para leer y escribir archivos YAML, que describiremos más abajo
- *pytz* [8]: Ayuda a gestionar los husos horarios de los eventos.
- *arrow* [9]: Simplifica el uso de fechas y horarios.
- *requests* [10]: Ofrece funcionalidad para hacer peticiones HTTP
- *calDAV* [11]: Cliente para conectarse a servidores CalDAV compatibles con la RFC 4791 [3]
- *iCalendar* [12]: Ofrece funcionalidad para gestión de calendarios y sus eventos, compatible con la RFC 5545 [13]
- *mkdocs* [14]: Ofrece funcionalidad para crear documentación del código a partir de los comentarios incrustados en el mismo
- *pytest* [15]: Ofrece funcionalidades para hacer tests del código desarrollado.

2.3 GIT

Git [16] es un software de control de versiones gratuito y de código abierto.

Permite almacenar un proyecto de desarrollo de código, documentos o ficheros, con control de cambios.

Permite guardar variaciones de código en lo que en Git se llaman *ramas,* versiones alternativas del código fuente que permite implementar parches o nuevas funciones del proyecto sin afectar a la versión principal hasta el momento de la fusión.

Git facilita la colaboración entre desarrolladores. Los repositorios de código se mantienen de forma distribuida, de manera que cada desarrollador tiene el repositorio completo con todas las versiones del proyecto que está en desarrollo. Para que todos los repositorios de código estén sincronizados, cada desarrollador puede sincronizar su repositorio con el de otro, o puede aceptar cambios de otro desarrollador manteniendo las variaciones que deseen.

Git tiene clientes para Android, iOS, Mac, Linux y Windows, tanto versiones consola como con interfaces gráficas.

Durante el desarrollo, vamos a utilizar Git con un repositorio remoto en GitHub, donde se mantendrá el código fuente. Dicho repositorio se encuentra en la url https://github.com/tfg-projects-dit-us/BotGuardianes

2.4 REST

REST [17] (Transferencia de Estado Representacional en español) es un estilo de arquitectura para sistemas distribuidos. Cuando un servicio web es conforme al estilo REST, se dice que ofrece una Interfaz de programación de aplicaciones (API) REST.

Funciona con un patrón de diseño de cliente-servidor. Consiste en dos tipos de componentes: clientes y servidores. El servidor espera a recibir peticiones de los clientes, y provee a éstos de un servicio.

Con este estilo de arquitectura, un cliente de una API REST puede solicitar un recurso, que será identificado de forma única en esta API, y manipularlo a través de dicha API.

Esto permite que un cliente, que requiere acceso a unos recursos, pueda conectarse con un servidor en remoto para obtener dicho recurso sin necesidad de almacenarlo localmente. Se simplifica con ello el mantenimiento y la portabilidad del cliente.

Para nuestro caso en particular, vamos a usar una API REST en desarrollo para acceder a los datos de los profesionales incluidos en el sistema de actividades, bien mediante su ID única que lo identifica, su correo o su ID de Telegram. Con estos datos podemos hacer lecturas y escrituras de información como el nombre y apellidos del usuario, su correo electrónico, o modificar la ID de Telegram del usuario.

2.5 API para bots de Telegram

La API REST de Telegram para bots [5], basada en HTTP, está diseñada para crear bots para Telegram de forma simplificada mediante clases y métodos que permiten el acceso a dicha interfaz.

Un bot es una aplicación que está a la espera de recibir peticiones, que procesará en segundo plano ofreciendo un resultado al usuario.

En Telegram, un bot tiene disponibles muchas funcionalidades similares a las de un usuario estándar, pero con ciertas limitaciones bien por seguridad o bien por decisiones de diseño. Por ejemplo, un bot puede comunicarse con un usuario mediante envío de mensajes y recepción de mensajes y comandos.

Sin embargo, no puede iniciar una conversación con un usuario a menos que éste comience la comunicación primero.

Se puede utilizar la funcionalidad de un bot en un chat grupal de Telegram o en un chat privado con el bot.

Un mensaje es una cadena de texto enviada a través de un chat o un canal de Telegram. Dicho mensaje tiene una ID única que lo identifica en un chat o canal. Y cada canal o chat tiene una ID única que lo identifica en el servicio de Telegram.

Un mensaje sólo puede ser editado o eliminado por el emisor del mensaje, y se puede crear en cualquier momento que el emisor quiera enviarlo. Estos mensajes se almacenan en los servidores de Telegram, y se envían a los receptores.

Entendemos por chat una conversación privada entre dos usuarios de Telegram. Esto incluye una conversación privada entre un bot y un usuario.

Un canal a su vez es un chat grupal especial, donde sólo pueden enviar mensajes los usuarios designados como administradores del canal.

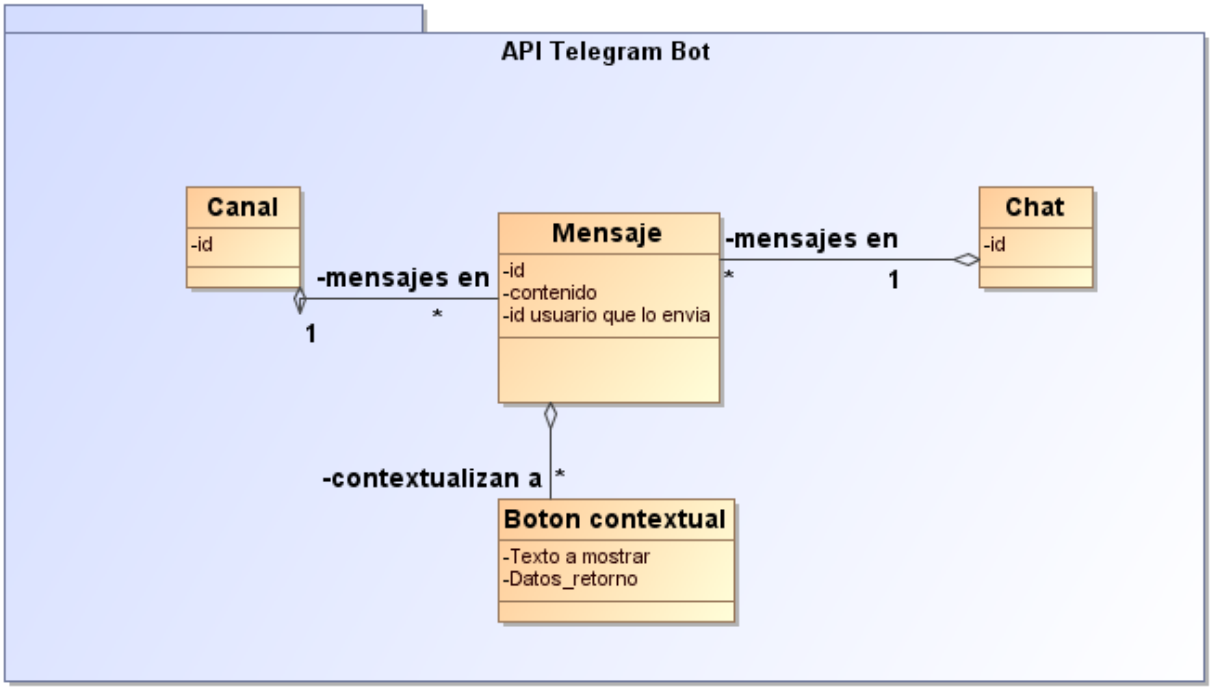


Ilustración 1: Relación botón contextual, mensaje, canal y chat

2.6 Docker

Docker [18] es un proyecto para facilitar la distribución de software utilizando contenedores. Un contenedor es, en esencia, similar a una máquina virtual, aunque reducida en utilización de recursos. Se utiliza como base del contenedor el kernel del sistema operativo residente, y el gestor de contenedores reparte los recursos necesarios a los contenedores actualmente en funcionamiento.

Los contenedores están aislados entre sí, y sólo utilizan los recursos que necesitan para mantener el software funcionando.

Esta independencia del sistema operativo residente permite que la distribución de software requiera menos esfuerzo en controlar las dependencias necesarias para su instalación. Además, permite su instalación en cualquier sistema operativo que soporte Docker.

2.7 iCalendar

Los calendarios utilizados para poder coordinar las asistencias utilizan el formato iCalendar [13]. iCalendar describe los eventos de un calendario con etiquetas para definir aspectos como el nombre del evento, la fecha de inicio y de fin, entre otros.

En este proyecto los eventos representan una actividad asistencial, y los participantes de la actividad se representan mediante los *asistentes* de un evento.

Los detalles que nos interesan de un evento son:

- Nombre de evento o Summary
- Fecha de inicio del evento
- Fecha de fin de evento
- Asistentes o attendees al evento. De los asistentes, obtenemos los datos siguientes
    - Correo electrónico del asistente
    - Rol del asistente
    - Tipo de asistente

Utilizaremos los calendarios generados por el proyecto *Gestión de calendarios para un servicio hospitalario* [2]. Para poder acceder a ellos, accederemos a un servicio donde se alojan descrito en el apartado “2.7.2 DAViCal”

2.7.1 CalDAV

CalDAV [3] es un protocolo utilizado para acceder y compartir calendarios con formato iCalendar. En este proyecto se utiliza para acceder a los calendarios donde se guardan los eventos que representan las actividades

2.7.2 DAViCal

DAViCal [19] es un servidor de código abierto que implementa el protocolo CalDAV para permitir alojar, acceder y compartir calendarios. Utiliza una base de datos PostgreSQL para almacenar la información de dichos calendarios.

En el proyecto se utiliza este servicio para los calendarios que utiliza esta aplicación.

3 REQUISITOS

Este capítulo describe los requisitos del proyecto. Han sido tomados a partir de analizar la situación actual según se describe en los proyectos "*Servicio para gestión de actividades asistenciales complementarias*" [1] y "*Gestión de calendarios para un servicio hospitalario*" [2]

Aunque hayan sido explicados en el capítulo de Introducción, vamos a describir formalmente los actores involucrados, los Casos de Uso y los requisitos, tanto funcionales como no funcionales.

3.1 Actores

En esta sección se definen los diferentes actores involucrados en el capítulo "1.1 Participantes identificados".

A-01	**Ofertante**
Descripción	Persona que desea cambiar una actividad, en la que actualmente es partícipe. Para cambiar dicha actividad ofrece su actividad para que otra persona la escoja.

Tabla 1: A-01 - Ofertante

A-02	**Demandante**
Descripción	Persona que desea participar en una actividad, previamente ofrecida por el ofertante. Esta persona solicita su participación en dicha actividad, a la espera de que le sea aprobado el cambio por el gestor.

Tabla 2: A-02 – Demandante

A-03	**Gestor**
Descripción	Persona que gestiona un servicio y la planificación de las actividades de servicio. Estará encargado, entre otras funciones, de autorizar los cambios acordados entre el demandante y ofertante

Tabla 3: A-03 – Gestor

A-04	**Participante**
Descripción	Persona que utiliza el servicio para todas las funciones, salvo las que utiliza el A-03 – Gestor. Los actores A-01 - Ofertante y A-02 – Demandante son subclases de este actor.

Tabla 4: A-04 - Participante

3.2 Caso de uso

3.2.1 Situación actual

- Intercambio de actividad

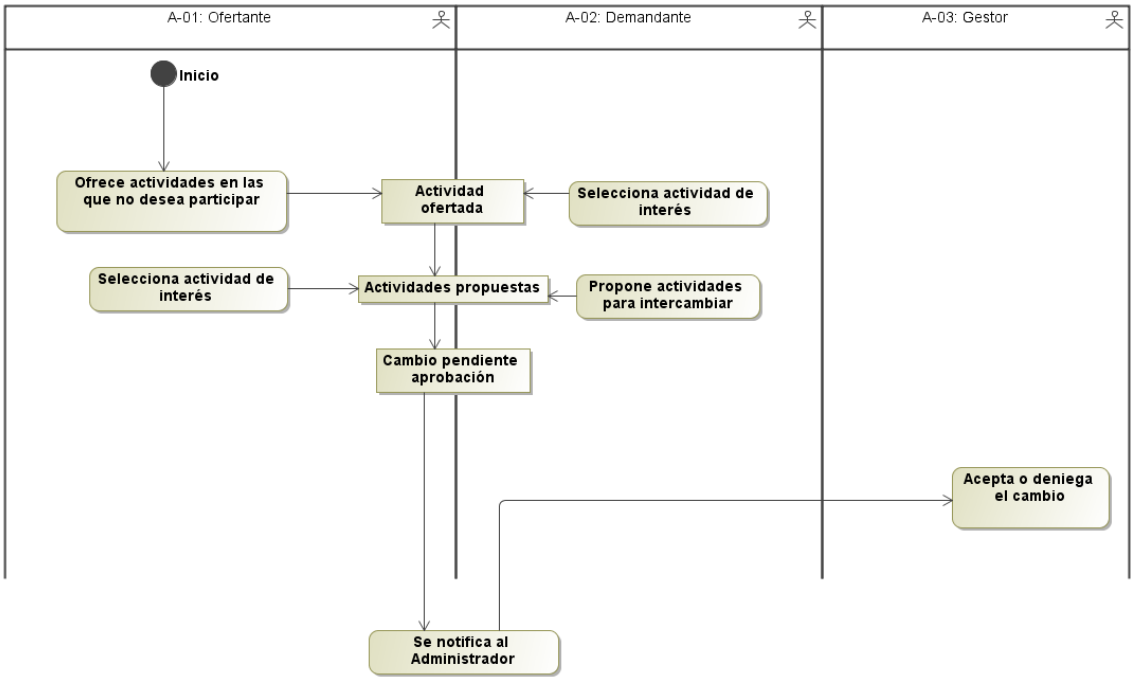


Ilustración 2: Diagrama actividad de intercambio de actividad, situación actual

Este proceso de intercambio de actividad se hace entre los compañeros conversando entre sí.

El ofertante ofrece una actividad que no desea, y pregunta por actividades que otros compañeros no desean. En cuanto un demandante solicita la actividad que está ofreciendo, el demandante propone actividades para intercambiar, y el ofertante elige una de su interés si la hubiera. Ambos actores aceptan el intercambio, y presentan una propuesta de intercambio de actividad al gestor.

Requiere de una búsqueda activa del ofertante hacia posibles demandantes, pues puede encontrar que no ofrezcan actividades de su interés

- Estados de la actividad

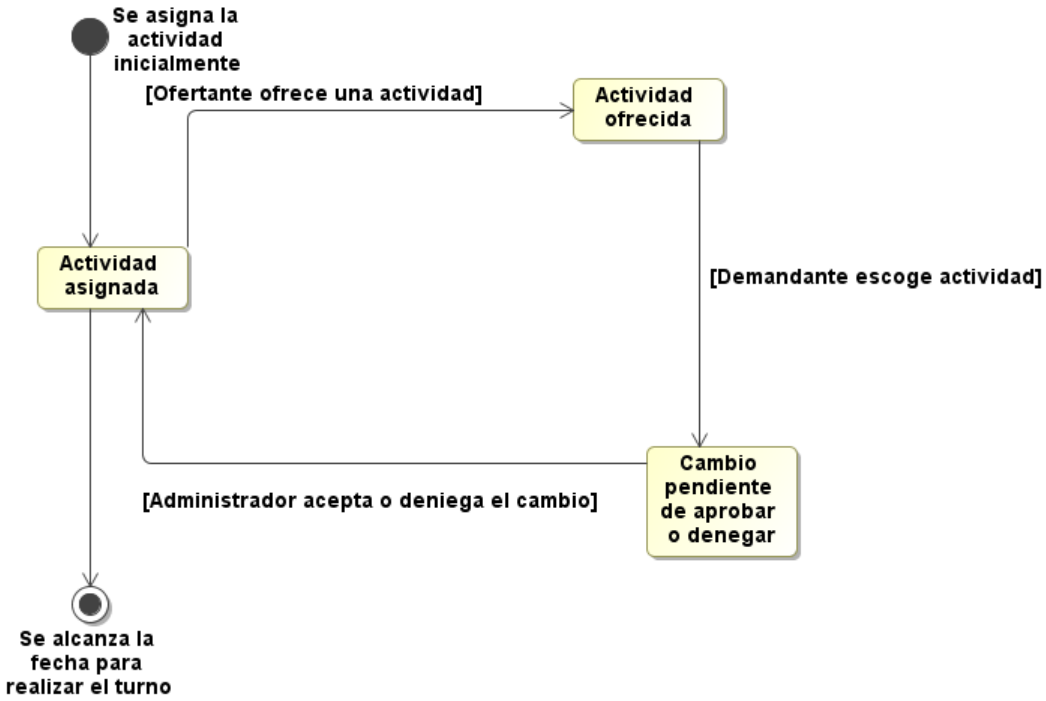


Ilustración 3: Estados de la actividad, situación actual

La actividad a efectos prácticos se ofrece y solicita simultáneamente. Es decir, el ofertante ofrece una actividad a sus compañeros pidiendo una de su interés. En cuanto alguno de sus compañeros puede ofrecerle una, elige alguna si le interesara. Ambos por tanto llegan a un acuerdo de intercambio y lo presentan al gestor para que se apruebe o deniegue.

Otra opción que tiene el ofertante es ceder una actividad. Consiste en ofrecer su actividad sin esperar una actividad propuesta por el demandante, de manera que en cuanto otro demandante solicita la actividad, se lleva el acuerdo de cesión al gestor para ser aprobado o denegado.

3.2.2 Situación deseada

Se incluye un diagrama de casos de uso para ilustrar qué funciones ofrece el bot de Telegram a los usuarios.

- Casos de uso

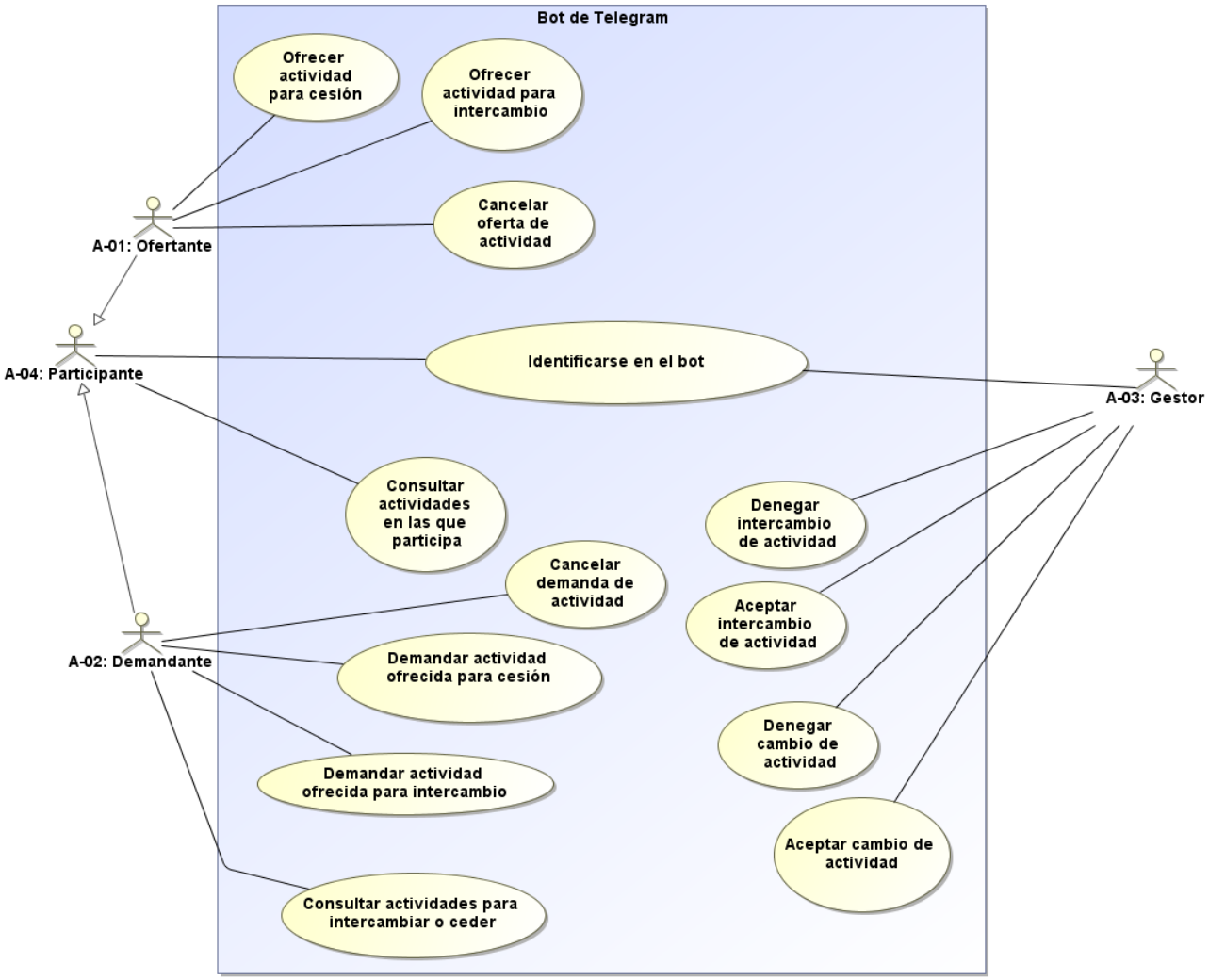


Ilustración 4: Diagrama de casos de uso, situación deseada

- Cesión de actividad

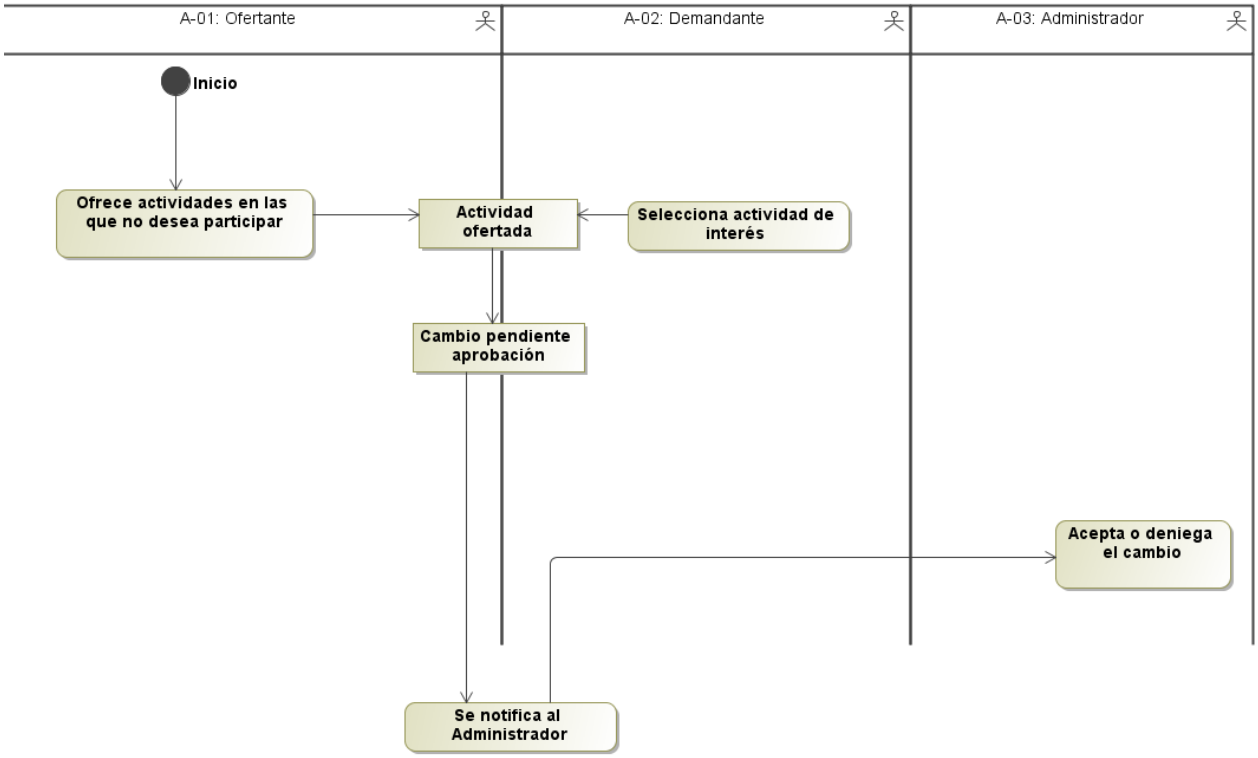


Ilustración 5: Diagrama actividad de cesión de actividad, situación deseada

El caso de cesión se diferencia con el intercambio en que el demandante no propone actividades para intercambiar. Sólo se cede la actividad a otro participante que la demande.

Para el caso de intercambio de actividad, el diagrama de actividad resultante sería igual que en “Ilustración 2: Diagrama actividad de intercambio de actividad, situación actual”, se mantiene el procedimiento, pero se ofrece soporte con el bot desarrollado

- Estados de cambio de actividad

Este nuevo diagrama de estados muestra que una actividad tiene la misma cantidad de estados que antes, ofreciendo la capacidad de cancelar la oferta de actividad al ofertante, mientras el A-03 – Gestor no apruebe o deniegue el cambio.

A su vez, el demandante también tiene la capacidad de cancelar la demanda mientras no se apruebe o deniegue el cambio.

Una vez la actividad está escogida por un demandante, se notifica al gestor para que se pueda aprobar o denegar.

Una vez aprobado o denegado el cambio, se convierte en una actividad asignada.

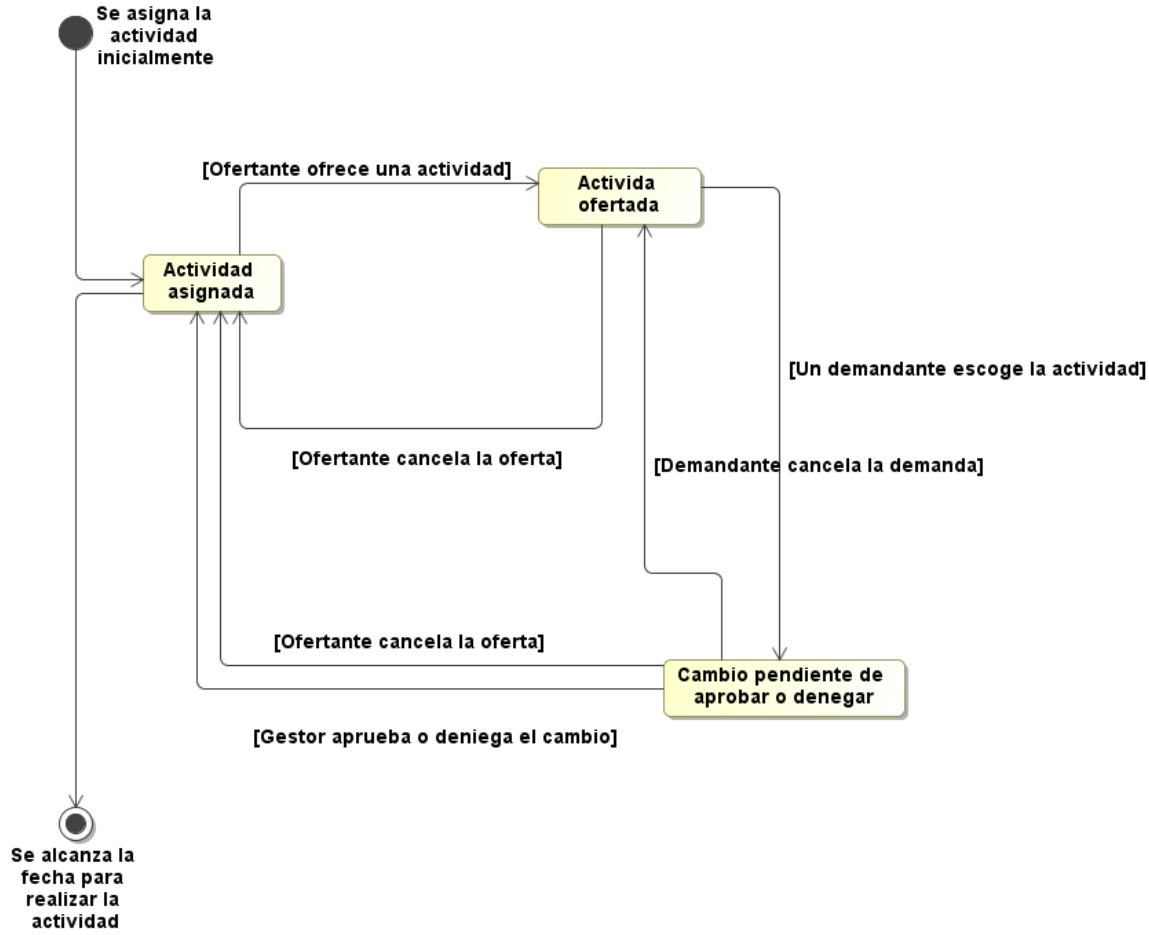


Ilustración 6: Estados de la actividad, situación deseada

3.3 Requisitos funcionales

Llamaremos a los requisitos funcionales RF

RF-01	**Identificación de usuario**
Descripción	Todos los usuarios de la aplicación deben identificarse en el bot. Utilizarán su correo electrónico como identificación única en él.

Tabla 5: RF-01 - Identificación de usuario

RF-02	**Iniciación de la aplicación**
Descripción	El usuario es quien inicia la aplicación y realiza la inscripción en la misma.

Tabla 6: RF-02 - Iniciación de la aplicación

3.4 Requisitos no funcionales

Llamaremos a los requisitos no funcionales RNF

RNF-01	**Comentarios de código**
Descripción	El código fuente del proyecto debe estar auto documentado con comentarios insertados en el código, siguiendo el esquema de Google docstring. Estos comentarios se pueden exportar para formar una web donde se mantenga la documentación

Tabla 7: RNF-01 - Comentarios de código

RNF-02	**Pruebas de unidad**
Descripción	El código fuente debe incluir pruebas de unidad para verificación de cambios de versiones con el funcionamiento completo.

Tabla 8: RNF-02 - Pruebas de unidad

RNF-03	**Trazas de código**
Descripción	La aplicación debe ofrecer trazas de código guardadas en ficheros para poder supervisar, depurar y mantener el programa, diferenciando por niveles de importancia y por el módulo que las genera.

Tabla 9: RNF-03 - Trazas de código

4 TRABAJO REALIZADO

En este capítulo se describe la aplicación desarrollada, que cubre los requisitos para facilitar el proceso de intercambio o cesión de actividades.

El código desarrollado se encuentra en https://github.com/tfg-projects-dit-us/BotGuardianes

Diferenciaremos el diseño estructural y de comportamiento de la aplicación.

También se detalla cómo se han cubierto los requisitos no funcionales.

4.1 Estructura

En este apartado ilustramos la arquitectura de la aplicación, de manera que se pueda entender cómo funciona cada componente y el conjunto.

El código fuente está dividido en ficheros llamados módulos, divididos por funcionalidad. Estos módulos pueden contener funciones y variables de módulo o también clases

- servicio_rest.py; Ofrece funcionalidad para comunicarse con la API REST de acceso a datos de los empleados
- telegram_tools.py: Utiliza la funcionalidad de la API de Telegram para bots, y ofrece la funcionalidad de la aplicación en los chats de Telegram.
- gestor_calendario.py: Ofrece funcionalidad para comunicarse con los calendarios que gestionan las actividades a la aplicación.
- config.py: Carga el fichero de configuración de la aplicación.
- MainBot.py: Módulo que opera con todos los anteriores para dar servicio a la aplicación.

Para ilustrar cómo se relacionan, se incluye un diagrama de clases

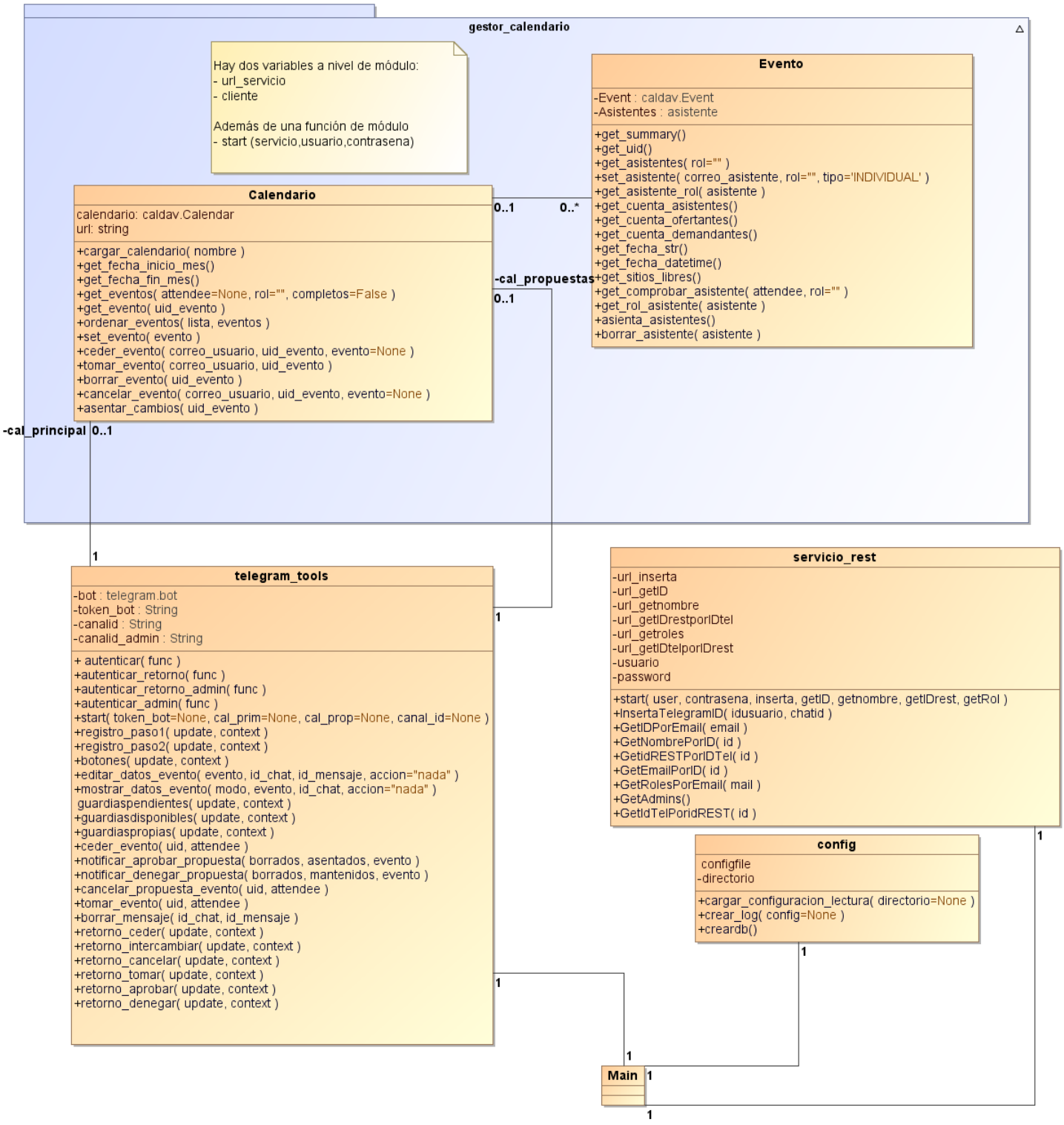


Ilustración 7: Diagrama de clases de la aplicación

En el diagrama, un evento del calendario representa una actividad asistencial. Ese evento se compone de:

- Resumen
- Descripción
- Fecha de inicio y de fin de la actividad
- Asistentes.

En el proyecto, utilizamos dos calendarios: El calendario de asignaciones (o calendario principal) y el calendario de propuestas. En el calendario de propuestas se hacen réplicas de los eventos del calendario principal para poder identificar las ofertas de actividades publicadas por los ofertantes.

Para este proyecto, el rol de asistente es lo que determina si está asignado o no actualmente a la actividad:

- Si es Requerido, es un asistente que tiene asignado dicha actividad actualmente
- Si es Opcional, el asistente es un Ofertante, que ha ofrecido esta actividad. Esto solo pasa en el calendario propuestas.
- Si es No Asistente, es un demandante que está solicitando la actividad. Este rol de asistente solo se utiliza en el calendario propuestas.

El tipo se utiliza para diferenciar una oferta para cesión o para intercambio.

En el software diferenciamos entre los *usuarios* participantes (A04) y el *administrador* (A03).

Un usuario participante (A04), tras autentificarse en la aplicación, tiene todas las capacidades del software, excluyendo las capacidades exclusivas del gestor, que en la aplicación lo hemos llamado administrador.

Un administrador también es un usuario, puede ser partícipe de las actividades asistenciales, ceder o intercambiar turnos, y demandar turnos ofrecidos. Además de las funciones de A-04 - Participante, tiene las capacidades de A-03 – Gestor de aceptar o denegar cesiones o intercambios propuestos por los participantes.

Para gestionar las acciones de los usuarios, se implementan botones. Hay 3 tipos de botones implementados:

- Botones contextuales de mensaje. Estos botones se sitúan debajo de un mensaje enviado por el bot bien a un chat con un usuario de Telegram, bien a un canal. Permiten acciones contextuales relacionadas con el mensaje. En este proyecto, estos botones se utilizan para incluir una acción a realizar en el contexto de un evento identificado con su uid.
    - Estas acciones son las necesarias para ofrecer las funcionalidades a los usuarios de la aplicación. La lista se compone de:
        - Ceder, para que el ofertante ofrezca una actividad en la que no desea participar, sin requerir otra a cambio.
        - Tomar, para que el demandante solicite la actividad cedida.
        - Cancelar, para cancelar la propuesta de cambio antes de que el administrador la deniegue o la apruebe
        - Intercambiar, para que el ofertante ofrezca una actividad en la que no se desea participar, requiriendo a cambio otra propuesta del demandante
        - Permutar, para que el demandante proponga una actividad a cambio de la ofrecida con la acción "Intercambiar"
        - Escoger, para que un ofertante pueda escoger una de las propuestas del demandante en el proceso de intercambio
        - Rechazar, para que un ofertante pueda rechazar todas las propuestas de un demandante.
        - Aprobar, para que un administrador apruebe un cambio
        - Denegar, para que un administrador deniegue un cambio.
- Botones de teclado de respuestas. Estos se utilizan para que los usuarios puedan enviar comandos al bot, sin esperar una acción previa del bot. Estos son:
    - Actividades propias, para consultar las actividades en las que el usuario participa. Al listar dichas actividades, añade botones contextuales, anteriormente presentados, con la acción ceder e intercambiar.
    - Actividades disponibles para solicitar cambio. Permite a un demandante consultar qué actividades están ofrecidas, ya sea cedidas o para intercambiar
    - Actividades pendientes de ser aprobadas o denegadas. Permite a un ofertante o a un demandante consultar las actividades pendientes de ser aceptadas o denegadas. Al listar dichas actividades, añade un botón contextual, anteriormente presentados, para poder cancelar dichas propuestas.
- Botones de comandos de Menú. Son necesarios para poder iniciar la aplicación la primera vez que se utiliza. Solo ofrece dos botones, el botón de inicio o "/start" y el botón de "/botones" para mostrar los botones de teclado de respuestas, para casos en los que no estén visibles dichos botones.

4.2 Comportamiento

En este apartado se muestra el funcionamiento del bot. Esto incluye las posibles interacciones del usuario, y una muestra del aspecto resultante. Además, comentaremos los casos de uso que ofrece la aplicación al A-04 – Participante y al A-03 – Gestor

4.2.1 Casos de uso

Se describen los casos de uso de la aplicación en los apartados siguientes.

4.2.1.1 Identificación de usuario

Todos los usuarios para poder acceder al servicio deben identificarse con su correo electrónico al iniciar conexión con el bot la primera vez que utiliza el servicio. Este caso de uso es necesario para adquirir la identificación de Telegram de un usuario ya existente en la plataforma. También deben ser agregados al canal de "Publicación de Actividades", para poder recibir las notificaciones de las actividades ofrecidas

Se incluye un diagrama de secuencia, mostrando cómo es la secuencia de identificación de un usuario. En el diagrama que se adjunta se muestra el caso de que la identificación sea exitosa.

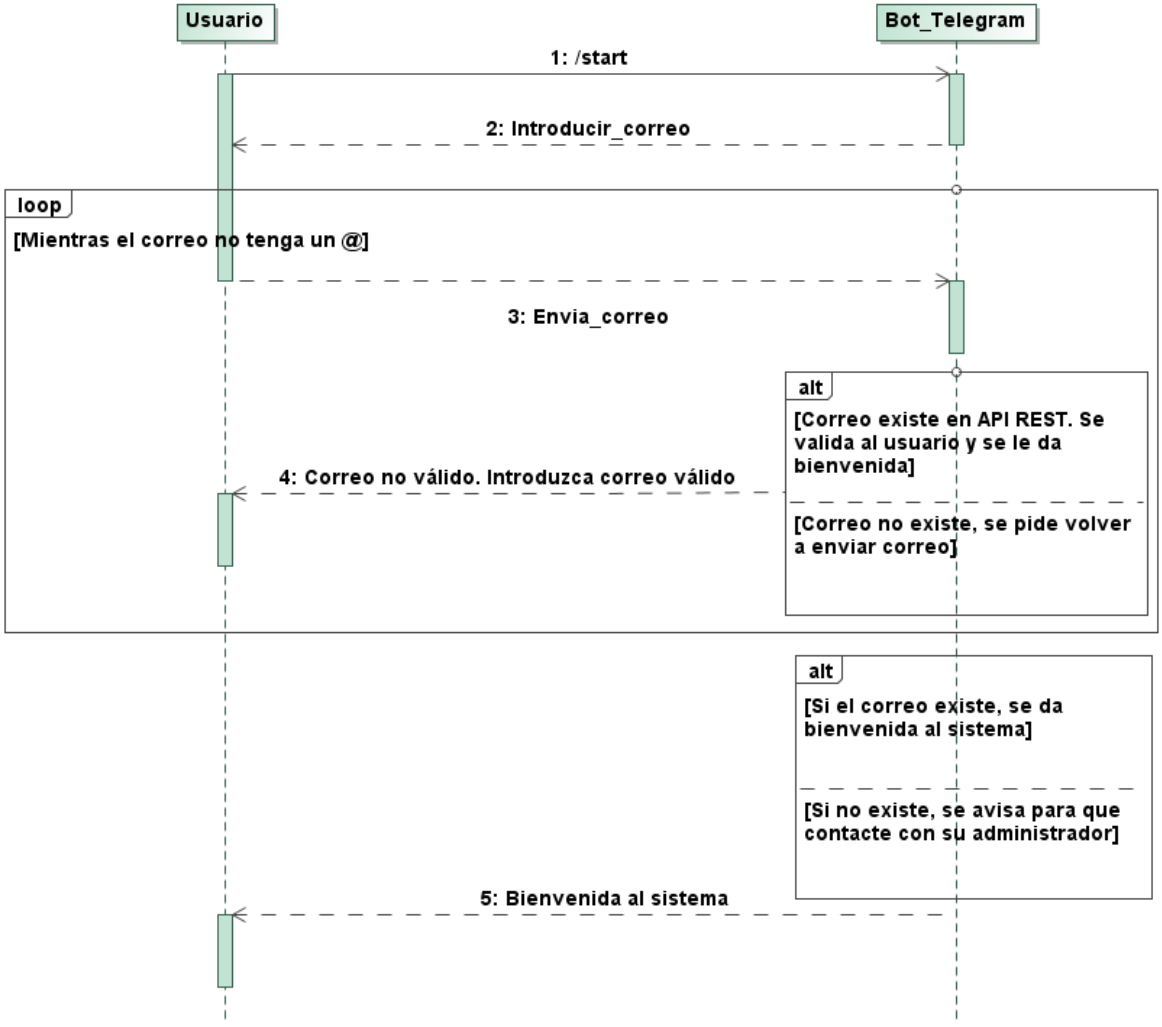


Ilustración 8: Secuencia de identificación en bot, situación deseada

Se incluye el aspecto del servicio en Telegram.

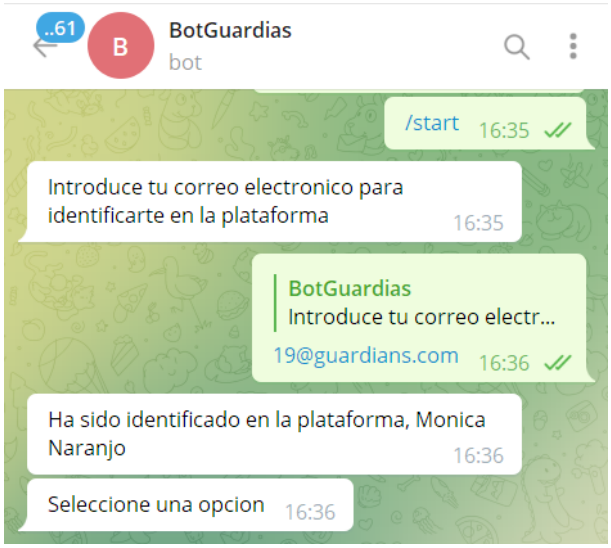


Ilustración 9: Interfaz de la aplicación en Telegram 1

En caso de que no se introduzca un correo existente en la plataforma, se avisará de ello y se solicitará que hable con el gestor del servicio para asegurar que su correo está correctamente configurado en la plataforma

Tras identificarse correctamente el participante, se mostrará en la sección inferior del chat un teclado con los botones de teclado de respuestas.

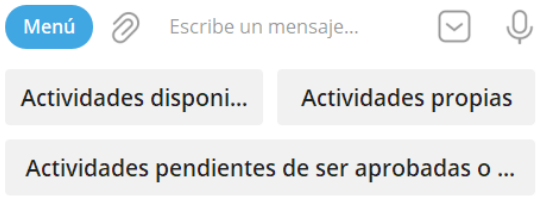


Ilustración 10: Teclado de respuestas

Para el caso de que el usuario sea administrador, se agrega un botón adicional debajo para mostrar los cambios disponibles para aprobar o denegar. Este usuario debe ser agregado al canal de Publicación Actividades para Administradores.

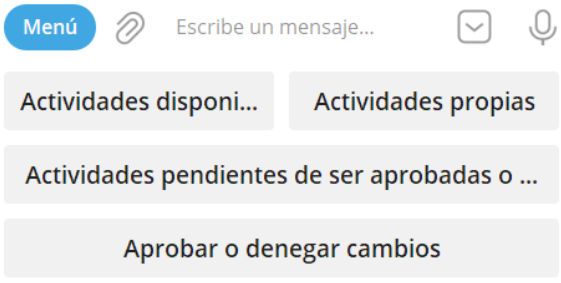


Ilustración 11:Teclado de respuestas de administrador

4.2.1.2 Consultar actividades en las que participa

Un participante puede solicitar las actividades en las que participa. Para ello, tras el registro, puede seleccionar el botón de "Actividades propias" y se le mostrará una lista de actividades en las que está actualmente participando, una por mensaje.

Esto desencadena que el bot llame a la función `telegram_tools.guardias_propias`. Esta función utiliza la función de `Calendario.get_eventos`, indicando el asistente que estamos buscando, que sería el usuario que ha hecho la consulta.

Este gestor de calendario devuelve los eventos encontrados en el calendario principal alojado en el servidor DAViCAL, y representa en un mensaje los datos de Nombre de evento, fecha de inicio, y los asistentes asignados a la actividad.

Al mostrarlas, añade un botón para poder ceder o intercambiar dichas actividades

Estos botones representan una acción y una id. La acción es lo que ejecutará el bot al pulsar el botón, y la id es la identificación única del evento.

En este caso, la acción es "ceder" o "intercambiar", según corresponda.

Se incluye una ilustración de la interfaz gráfica en Telegram

Ilustración 12:Interfaz de Telegram, Participante consulta actividades propias

4.2.1.3 Ofrecer actividad para cesión

Un ofertante una vez consulta las actividades en las que participa, puede ceder esta actividad.

Para ello, se replica el evento, se marca al asistente como opcional o *'OPT-ATTENDEE'* y se añade al calendario de propuestas. Así se reutiliza un estado estándar para representar que este participante quiere ceder su participación.

Una vez hecho esto, se enviará un aviso en el canal de "Publicación de Actividades" para que otros participantes puedan demandar esta actividad. En este mensaje se añade un botón para poder solicitar esta actividad, con la acción "tomar".

El id del mensaje en el canal de publicaciones se guarda en una base de datos sqlite para poder editar o eliminar el mensaje en otros casos de uso.

También se elimina el mensaje relacionado con el evento ofrecido para ceder, para que no pueda volver a proponer una cesión o intercambio.

A partir de aquí, si intenta consultar las "Actividades Propias", ésta aparece, pero no da la opción de ceder la actividad ni intercambiarla.

Se muestra debajo el aspecto que tiene la interfaz gráfica cuando un ofertante cede una actividad

Ilustración 13: Interfaz de Telegram, ofertante cede una actividad con éxito

4.2.1.4 Consultar actividades para intercambio o cesión

Una vez el ofertante ha ofrecido su actividad, otros participantes pueden demandarla para participar en su lugar.

Para este caso de uso, el demandante pulsaría en el chat con el bot en el botón de “Actividades disponibles para solicitar cambio”

Al pulsarlo, el bot envía los eventos que están en cesión o intercambio, agregando un botón contextual con la acción “tomar” en ambos casos para poder solicitarla.

4.2.1.5 Demandar actividad ofrecida para cesión

El demandante puede solicitar las actividades ofrecidas para cesión.

Para ello, pulsaría en el botón de “Pedir esta actividad cedida” en el canal de “Publicación de Actividades”, o bien, pulsando el botón de “Pedir esta actividad cedida” al ejecutar el caso de uso 4.2.1.4 “Consultar actividades para intercambio o cesión”. Tras hacer esta demanda de actividad, el mensaje en el canal de “Publicación de Actividades” relacionado con esta actividad se elimina, utilizando la id de mensaje almacenado en la base de datos sqlite.

Una vez hecho esto, se inscribe al demandante en el Evento como No Asistente o *“NO-ATTENDEE”.*

Si se diera el caso de que varios demandantes demandan la misma actividad, la primera que llega al bot es la tomada por válida, mientras que a la otras se les avisa de que no es posible demandar ésta por no tener huecos libres.

En este punto, se informa a los administradores con un envío al canal de “Publicación Actividades para administradores”.

Se añaden ilustraciones mostrando el aspecto de la interfaz gráfica, desde el punto de vista del demandante

Ilustración 14: Publicación de Actividad cedida en el canal de Telegram

Ilustración 15: Actividad en la que se ha inscrito el demandante

4.2.1.6 Aceptar cesión de actividad

En este caso de uso, el Administrador puede aceptar la cesión propuesta por el ofertante y el demandante.

Para ello, una vez recibe la notificación en el canal de "Publicación Actividades para administradores", puede pulsar el botón de "Aprobar este cambio de actividad" para aceptar este cambio. Esto desencadena la acción de "aprobar".

Al hacer esto, el evento en el calendario de propuestas transfiere los datos al evento en el calendario principal, y acto seguido, se elimina el evento en el calendario de propuestas.

Además, se notifica a los afectados en el cambio de que el cambio se ha realizado.

Se adjunta una captura para mostrar el aspecto de la interfaz gráfica desde el punto de vista del administrador

Ilustración 16: Publicación de la propuesta de cesión en canal de administradores

Se incluye un diagrama de secuencia de un cambio de actividad completo, cuando un ofertante la cede.

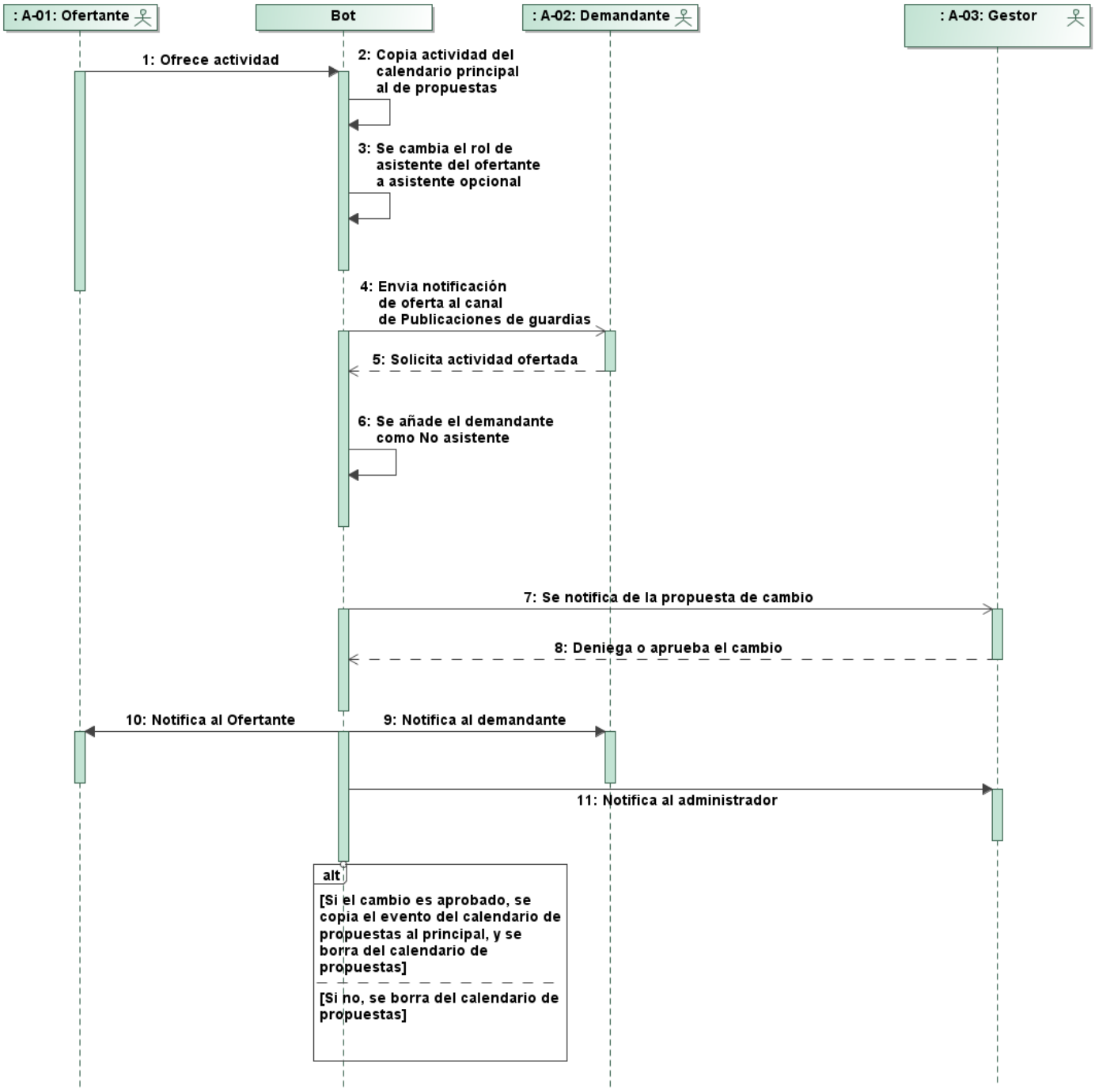


Ilustración 17: Diagrama de secuencia de cesión de actividad, situación deseada

4.2.1.7 Denegar cesión de actividad

En este caso de uso, el Administrador no admite la cesión propuesta.

En este caso, se notifican a los usuarios afectados en la cesión, y se elimina el evento en el calendario de propuestas, sin hacer cambios en el evento del calendario principal.

4.2.1.8 Ofrecer actividad para intercambio

Un ofertante tiene la capacidad de ofrecer una actividad, esperando que el demandante proponga una actividad o varias propias para permutarla con la actividad ofrecida.

El cómo se implementa es igual que la cesión, excepto que la acción que agrega al botón contextual en el canal de “Publicación de Actividades” es la de “permutar”.

Se aporta un diagrama de secuencia para ilustrar el proceso completo de intercambio.

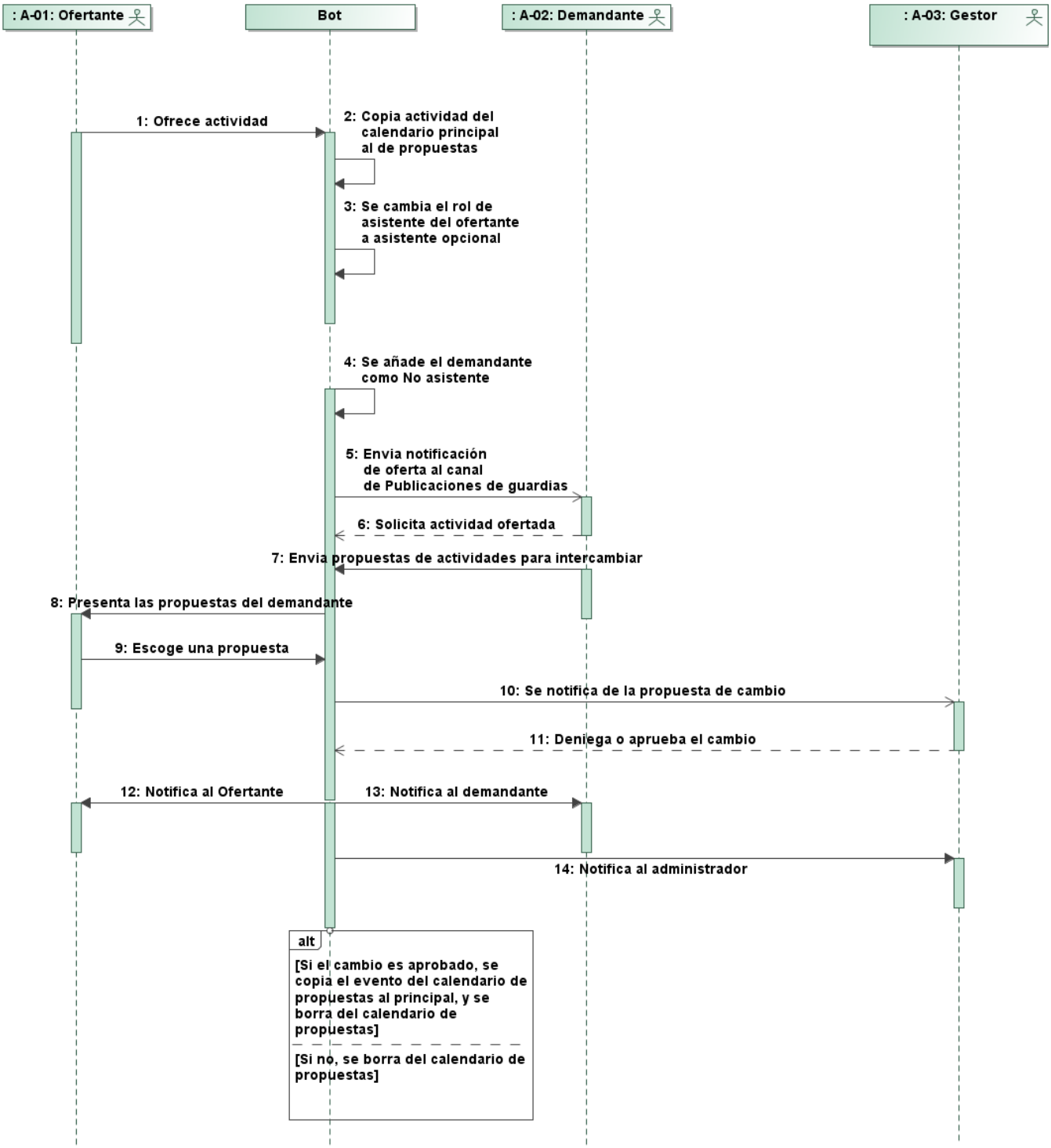


Ilustración 18: Diagrama de secuencia de intercambio de actividad, situación deseada

4.2.1.9 Demandar actividad ofrecida para intercambio

En este caso, una vez la actividad está ofrecida, el demandante solicita dicha actividad, ofreciendo a cambio como propuestas otras actividades propias para dicho intercambio.

El ofertante recibirá estas propuestas, una por mensaje, con botones contextuales con la acción “escoger”. Además, también tendrá la posibilidad de rechazar las propuestas con un botón con la acción “rechazar”.

4.2.1.10 Aceptar intercambio de actividad

El administrador puede aceptar el intercambio de actividad propuesto en cuanto se complete el proceso.

Una vez que se acepte el intercambio de actividad, de forma similar al caso de uso 4.2.1.6 “Aceptar cesión de actividad”, se establecen los datos de cada evento en el calendario principal, se elimina el evento del calendario de propuestas, y se notifica a los participantes afectados.

4.2.1.11 Denegar intercambio de actividad

El administrador también puede denegar el intercambio de actividad propuesto. De igual forma que en el capítulo 4.2.1.7 “Denegar cesión de actividad”. Al final, se elimina el evento generado en el calendario de propuestas y se notifica a los participantes afectados.

4.2.1.12 Cancelar oferta de actividad

Mientras el Gestor no acepte o deniegue una cesión o un intercambio, un ofertante puede cancelar la propuesta de actividad.

Para ello, pulsaría en el botón de teclado de respuestas “Actividades pendientes de ser aprobadas o denegadas”. Esto desencadenaría que el bot muestre en el chat, un evento por mensaje, las actividades que están a la espera de que el administrador las apruebe o deniegue. Será posible cancelar la oferta siempre que el Administrador no haya hecho una aceptación o denegación.

Al final de la cancelación, se enviarían notificaciones a los afectados.

Se incluye diagrama de secuencia de la cancelación de una oferta

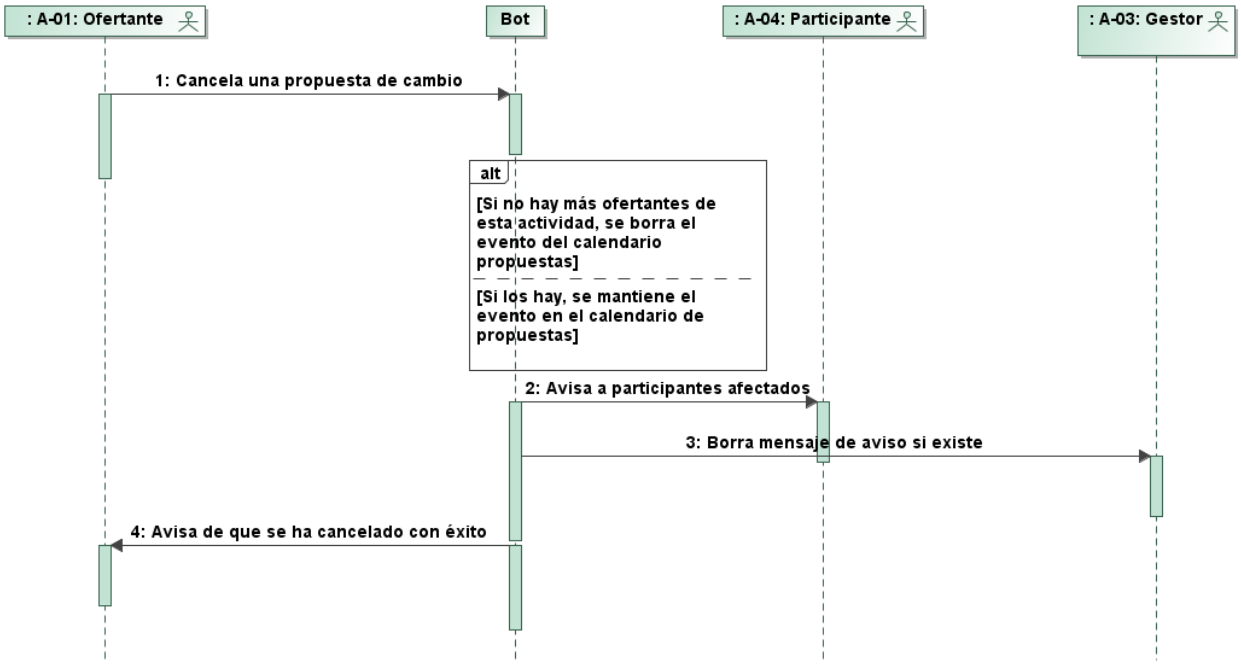


Ilustración 19: Cancelación de una oferta

4.2.1.13 Cancelar demanda de actividad

Mientras el Gestor no acepte o deniegue una cesión o un intercambio, un demandante puede cancelar la propuesta de actividad.

Para ello, pulsaría en el botón de teclado de respuestas “Actividades pendientes de ser aprobadas o denegadas”. Esto desencadenaría que el bot muestre en el chat, un evento por mensaje, las actividades que están a la espera de que el administrador las apruebe o deniegue. Será posible cancelar la demanda siempre que el Administrador no haya hecho una aceptación o denegación.

Al final de la cancelación, se enviarían notificaciones a los afectado.

Se incluye diagrama de secuencia de la cancelación de una demanda.

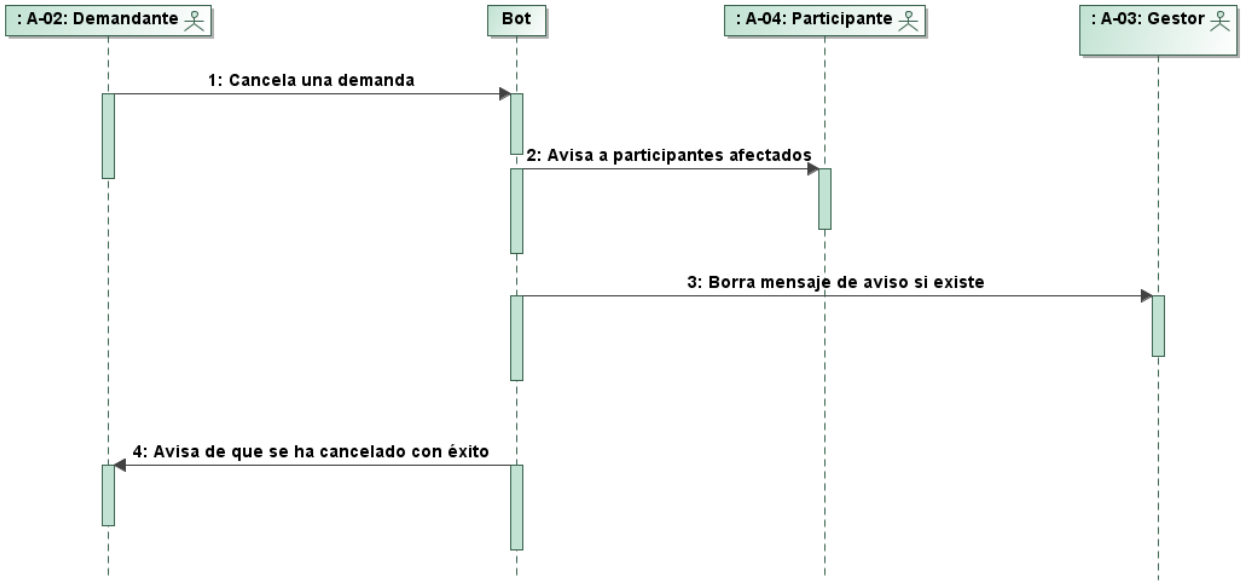


Ilustración 20: Cancelación de demanda

4.3 Pruebas de unidad

Para cumplimentar el requisito “RNF-02 - Pruebas de unidad”, se han creado tests de unidad utilizando el módulo pytest [15] y su plugin pytest-check [20].

Este módulo es diferente al de unittest, el módulo incluido con la instalación por defecto de Python, ya que permite hacer tests sin crear clases para ello. Solo es necesario crear funciones con la palabra “test_” al principio, y se considerará un test. Su implementación es más sencilla que la de unittest y en cuanto un test falla, muestra dónde ha fallado, comparando el resultado esperado con el resultado recibido.

El fichero donde se crean dichas funciones también debe contener como primera palabra “test_”. Esto permite que se creen múltiples ficheros con distintas pruebas de unidad, sin tener que cargarlos en ninguna plantilla.

Para ejecutar una prueba de unidad, se ejecuta la función que se pretende comprobar, y se guarda su resultado. A su vez, se almacenan los datos que se esperan como resultados en variables, y se comparan ambos. Para comparar, pytest usa las instrucciones “assert”, que comprueba la condición que se especifica. Normalmente la condición es de igualdad, pero se pueden utilizar otras.

El assert en cuanto encuentra un error en la comparación, detiene la ejecución de una prueba concreta. Al finalizar la ejecución, aporta un resumen de las comprobaciones que haya hecho en todos las pruebas.

Se presenta un ejemplo de código de un test de pytest

```
def test_prueba():
    resultado_esperado=X
    resultado=función(parámetro)


    assert resultado == resultado_esperado
```

Código 1: Código ejemplo test

También se muestra el resultado de un test completado con éxito, y otro con fallo.

```
==============================test session starts ==============================
collecting ... collected 1 item

test_prueba.py::test_prueba PASSED                                    [100%]

============================== 1 passed in 0.10s ===============================
```

```
test_prueba.py::test_prueba FAILED                                    [100%]
test_prueba.py:2 (test_prueba)
5 != X

Expected :X
Actual   :5
<Click to see difference>

def test_cuenta():
>       assert 5 ==X
E       assert 5 == X

test_assert.py:5: AssertionError
```

Ilustración 21: Resultado de test con éxito y con error

Para evitar el problema de que se detenga en mitad de una prueba, utilizamos el plugin pytest-check. Éste permite hacer varias comprobaciones con los métodos que aporta su clase principal, y en caso de fallar uno de ellos, continúa comprobando los demás.

4.4 Comentarios de código

Para poder facilitar el futuro desarrollo de la aplicación, se ha documentado el código usando el estilo de Google [21].

La guía de estilo básica es escribir entre tres pares de comillas dobles el contenido del comentario cuando comenzamos a comentar un módulo, una función, una clase o un método. La primera línea contiene la descripción del contenido de lo que se está comentando.

Después se hace una descripción más larga y se añaden algunos campos para explicar secciones de lo que se comenta.

Para casos de métodos o funciones, se pueden utilizar los siguientes

- "Args" para describir argumentos de entrada.
- "Returns" para retornos de salida
- "Raises" para comentar excepciones que puede devolver

En casos de clases, podemos usar "Atributes" para comentar los atributos de una clase.

Se añade un comentario de ejemplo con el código siguiente

```
def función_ejemplo(param1,param2):
"""
Descripción resumida de la función

Descripción más larga en varias líneas

Args:
    param1: Descripción del parámetro 1
    param2: Descripción del parámetro 2

Returns:
    Descripción de los valores que se devuelve

Raises:
    NombreExcepcion: Descripcion de la excepción

"""
```

Código 2: Función de ejemplo comentada con la guía de estilo de Google

Una vez hechos los comentarios en el código siguiendo este formato, transformamos todos los comentarios a una página web con el módulo mkdocs.

Esta documentación se encuentra alojada en https://tfg-projects-dit-us.github.io/BotGuardianes/

En el Anexo B – Generar documentación se explica cómo generar la documentación y subirla al repositorio de código

4.5 Distribución de la aplicación

Para distribuir la aplicación, se va a utilizar un contenedor Docker. Un contenedor se construye a partir de una imagen definida en un Dockerfile, un fichero que define la imagen base, y qué instrucciones debe ejecutar para preparar el contenedor listo para distribuirse.

El aspecto del Dockerfile es el siguiente:

```
FROM ubuntu:22.04

ADD ./modulos /BotGuardianes/modulos
ADD ./MainBot.py /BotGuardianes
ADD ./requirements.txt /BotGuardianes
ADD ./config_ejemplo /BotGuardianes/config_ejemplo
RUN mkdir /BotGuardianes/data
WORKDIR /BotGuardianes
RUN apt update
RUN apt upgrade -y
RUN apt install python3.10 pip gettext-base language-pack-es -y
RUN python3.10 -m pip install --no-cache-dir -r requirements.txt
```

Código 3: Dockerfile para distribución de aplicación

Con FROM se indica desde qué imagen base se parte para crear el contenedor actual.

ADD permite añadir carpetas y su contenido en un directorio concreto del contenedor

RUN permite ejecutar instrucciones dentro del contenedor durante la creación del mismo.

WORKDIR permite seleccionar el directorio de trabajo cuando se inicia el contenedor.

4.5.1 Dockerhub

Para distribuir el contenedor a través de Dockerhub, es necesario seguir la guía dispuesta en la web de Docker Repositories on Dockerhub

En resumen, se debe crear una cuenta en Dockerhub, crear un repositorio remoto y hacer inicio de sesión en la sesión local de Docker. Luego se puede construir el contenedor con la instrucción

```
docker build -t <hub-user>/<repo-name>[:<tag>] PATH
```

Siendo <hub-user> el usuario que tengamos en Dockerhub, <repo-name> el repositorio donde se aloje el contenedor, y <tag> una etiqueta opcional para referirnos a la versión de software. PATH es la dirección donde se almacena el Dockerfile necesario para construir el contenedor.

Luego podemos almacenarlo en dockerhub con

```
docker push <hub-user>/<repo-name>:<tag>
```

4.6 Despliegue de la aplicación

El despliegue de la aplicación no requiere más que tener docker instalado en el sistema.

Para hacer el despliegue se requiere descargar el fichero “docker-compose.yml” del [Repositorio de BotGuardianes en GitHub (https://github.com/tfg-projects-dit-us/BotGuardianes)](https://github.com/tfg-projects-dit-us/BotGuardianes) y seguir las instrucciones de “Producción” en el Readme del repositorio.

En dichas instrucciones se explica cómo configuraremos las variables de entorno que se encuentran en el fichero con los valores correctos.

Las instrucciones detalladas pueden consultarse en [ésta guía para instalar Docker en la máquina y docker-compose (https://docs.docker.com/compose/install/linux/#install-using-the-repository)](https://docs.docker.com/compose/install/linux/#install-using-the-repository).

Solo faltaría ejecutar con privilegios de administrador `docker compose up -d` para iniciar la aplicación

Se incluye un diagrama de componentes para indicar la relación de componentes entre sí.

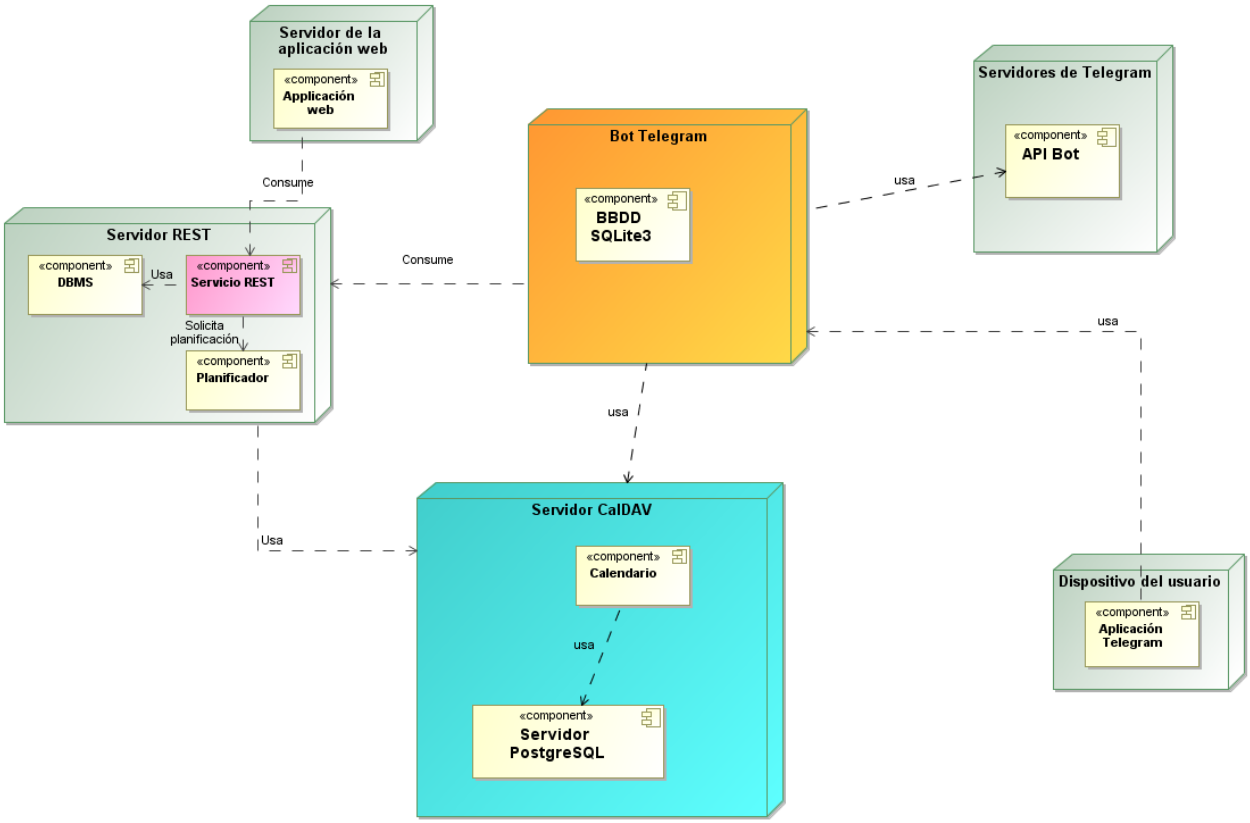


Ilustración 22: Componentes del servicio

5 LÍNEAS DE CONTINUACIÓN Y CONCLUSIONES

En este capítulo se pretenden detallar las líneas de continuación en el desarrollo de la aplicación. Se realiza además un resumen del trabajo.

5.1 Agregar funcionalidad para reservar citas

La aplicación desarrollada permite cambiar actividades entre los profesionales sanitarios. Un posible desarrollo a partir de ésta sería permitir reservar citas con un profesional. Un usuario del servicio sea sanitario o de otro tipo, podría solicitar una cita con el profesional y así reservar su tiempo. Dado que ya se trabaja con eventos de calendario, añadir la funcionalidad necesaria para reservar tiempo podría utilizar código ya existente

5.2 Añadir envíos de cambio del calendario a la API REST

La aplicación actualmente interacciona con el calendario alojado en el servidor CalDAV, pero no envía los eventos modificados a la API REST, de manera que aún no quedan reflejados en la aplicación de planificación de actividades. Para ello, habría que extraer los eventos en el momento que se aprueban, y enviarlos a la API REST con el formato iCalendar adecuado. Esta función actualmente está en desarrollo.

5.3 Simplificación de configuración

La configuración actual requiere añadir muchas url, como se puede ver en la guía de "Producción" de Repositorio de BotGuardianes en GitHub (https://github.com/tfg-projects-dit-us/BotGuardianes), es posible reducir dicha configuración, bien obteniendo las URL del servidor donde se aloje la API REST, o desde una base de datos.

5.4 Conclusiones

El proyecto se enfoca en añadir nuevas funcionalidades al proyecto "*Servicio para gestión de actividades asistenciales complementarias"* [1] y al proyecto "*Gestión de calendarios para un servicio hospitalario*" [2] se permite a los participantes intercambiar sus actividades asistenciales desde un teléfono móvil, utilizando los servicios de Telegram.

A la conclusión del trabajo, la funcionalidad está implementada mediante un bot de Telegram que, utilizando la API REST de los proyectos anteriores, permite la conectividad con el sistema ya existente.

Además, se conecta al servidor DAViCal para obtener el calendario de actividades asignadas, creando adicionalmente otro calendario para guardar las propuestas de cambio.

Para facilitar el despliegue de la aplicación, se proporciona la aplicación empaquetada en un contenedor Docker. También se proporciona el dockerfile para poder construir la imagen del contenedor y el docker-compose.yml para indicar las tareas que Docker necesita ejecutar para poner la aplicación en funcionamiento.

6 ANEXOS

Anexo A - Guía de activación de una máquina virtual en Azure

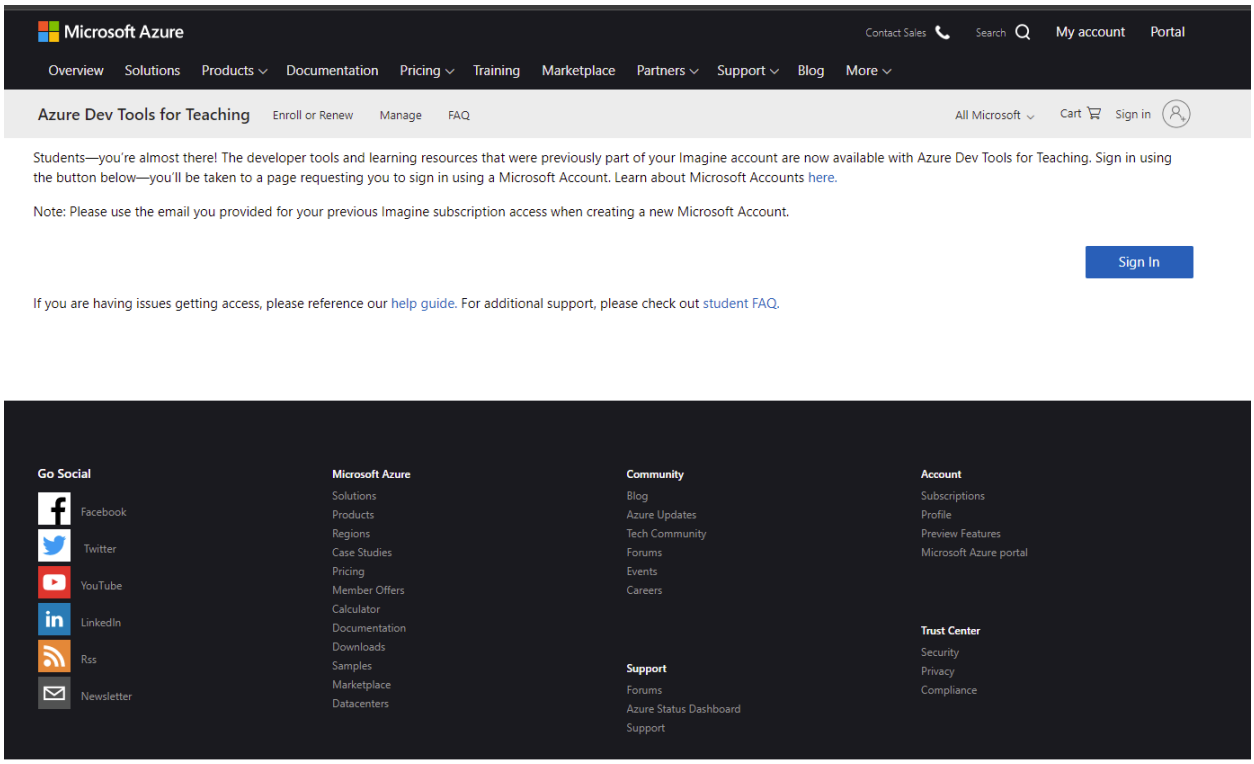
Para el alojamiento del bot, se va a utilizar una máquina virtual de Azure con Ubuntu Server.

Al tener una cuenta universitaria en la Universidad de Sevilla, disponemos de muchos servicios que provee Microsoft, que nos permiten licencias de uso de software y alojamiento en la nube, entre otras

Podemos ver los servicios que incluye la cuenta gratuita de estudiantes de Azure en [Azure para estudiantes: preguntas más frecuentes de Azure | Microsoft Azure (https://azure.microsoft.com/es-es/free/free-account-students-faq/)]

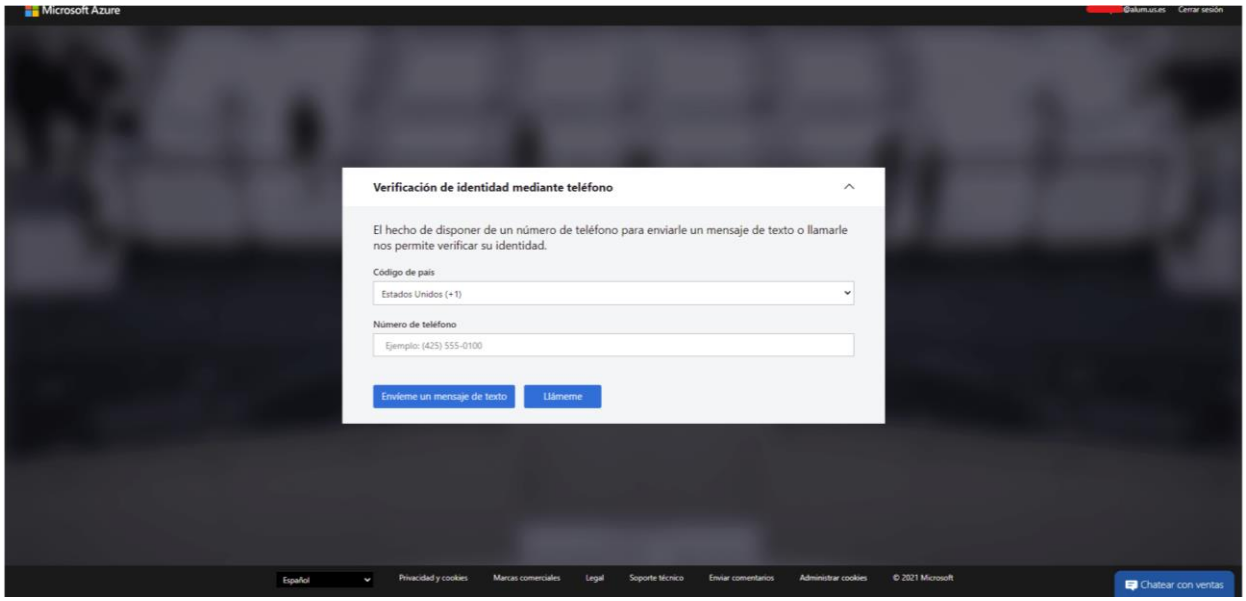
Este anexo pretende ser una explicación de cómo activar la cuenta de la Universidad de Sevilla para poder utilizar los servicios de Azure y crear una máquina virtual en línea.

Para poder activar el servicio incluido en la cuenta, tenemos que entrar en [Azure for Education (https://azureforeducation.microsoft.com/devtools)] y deberemos hacer inicio de sesión pulsando en el botón "Sign In" con las credenciales de la US, usando el correo @us.es o @alum.us.es según corresponda. Automáticamente nos llevará al Inicio de Sesión Único de la US donde podremos iniciar sesión con nuestras credenciales habituales.



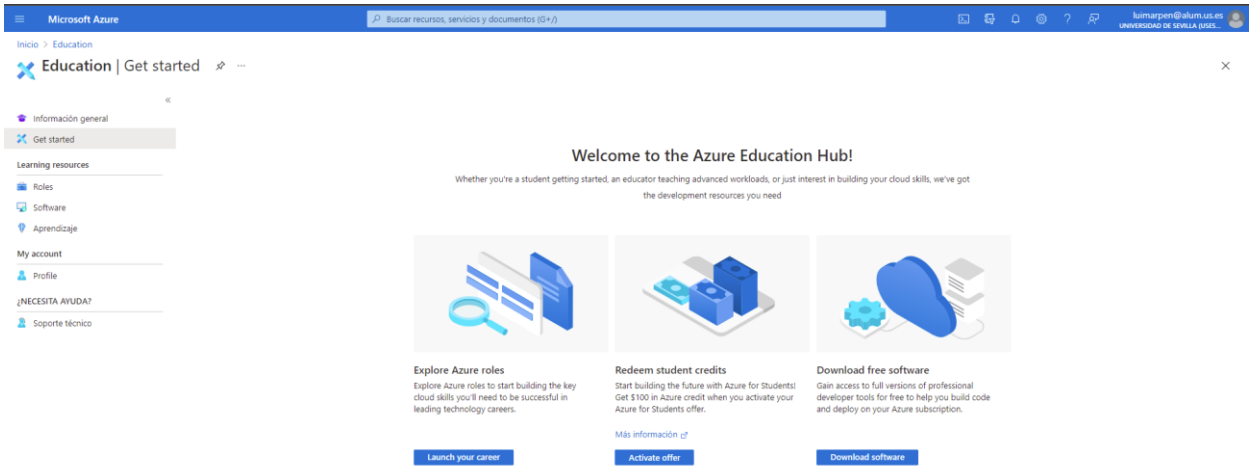
Anexo A - Ilustración 23: Azure For Education, inscripción al servicio

Tras hacer inicio de sesión, la primera vez nos solicitará un número de teléfono de verificación para asegurar que somos una persona real. Deberemos proporcionar un teléfono al que poder contestar una llamada o recibir un mensaje de texto, y tras validarlo, podremos continuar.



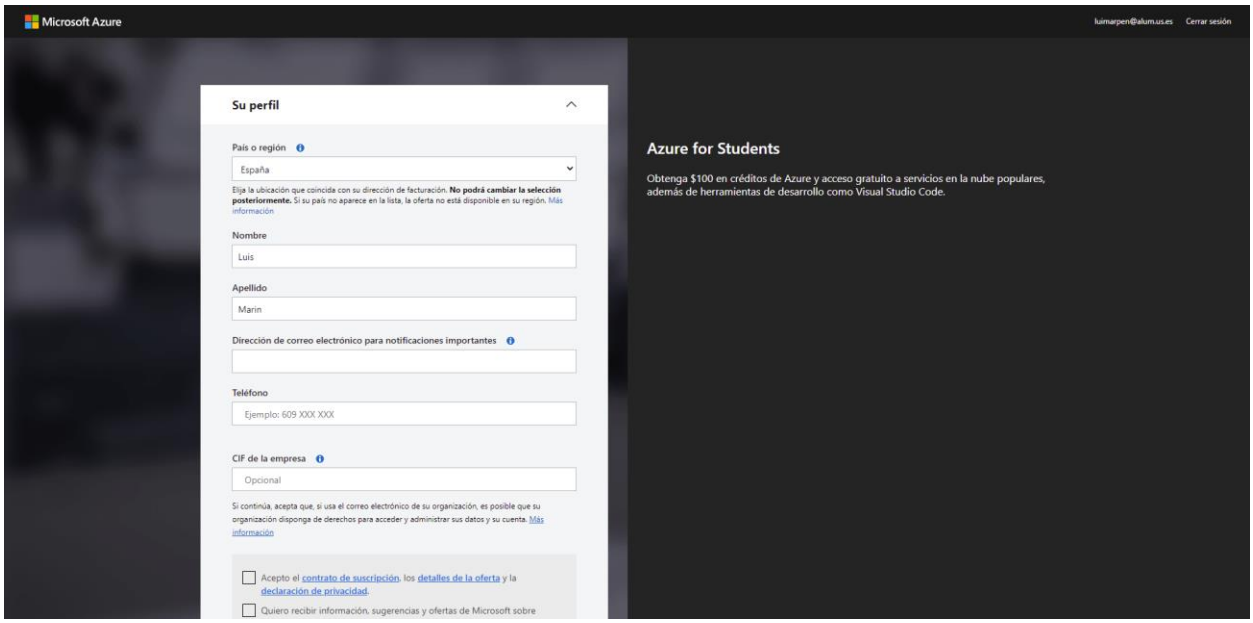
Anexo A - Ilustración 24: Azure for Education, verificación de identidad

Una vez introducido el teléfono podemos pedir que o bien nos llame por teléfono o nos envíe un SMS. Tras verificar el teléfono, tendremos acceso al HUB de Microsoft Azure



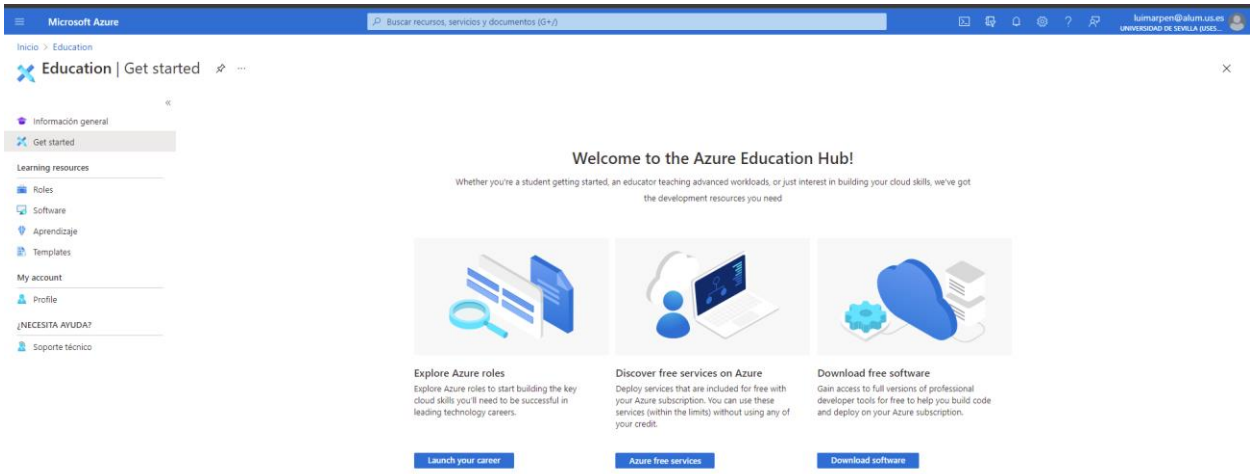
Anexo A - Ilustración 25: Azure for Education: Portal principal

El siguiente paso sería pulsar en “Activate Offer” para activar los créditos gratuitos de alumno. Esto requiere añadir un teléfono y un correo electrónico de avisos importantes



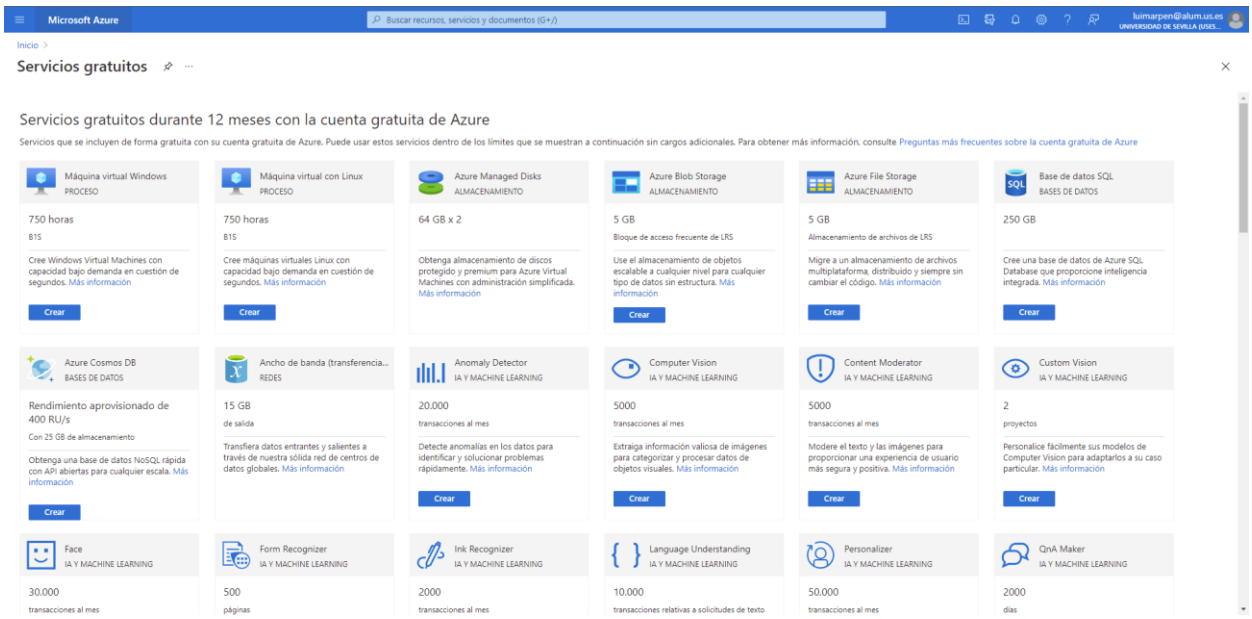
Anexo A - Ilustración 26: Activación del perfil de Azure For Education

Una vez terminado este paso, podremos entrar en los Azure Free Services



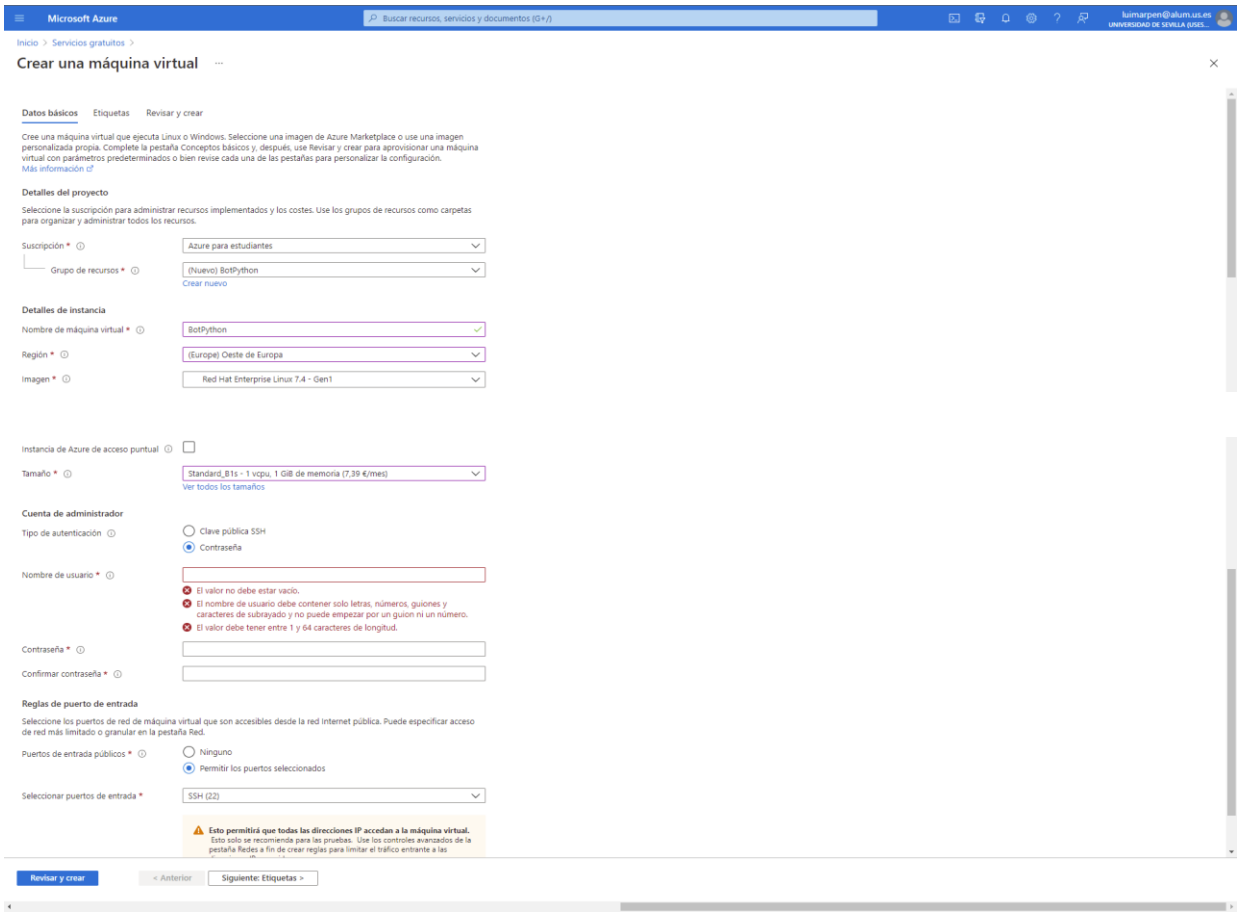
Anexo A - Ilustración 27: Azure Free Services

Necesitaremos una máquina virtual. Podemos usar tanto una con Linux como con Windows. Como servicio gratuito (Que no gasta crédito de la cuenta) se incluye una MV con 1 núcleo, 1GB de RAM y 4GB de almacenamiento persistente. Pero también se dispone de 100$ de crédito de Azure (Que caducan a los 12 meses), por lo que se puede hacer uso de las máquinas virtuales de pago con el crédito que se incluye.



Anexo A - Ilustración 28: Azure Free Services, detalle de los servicios

En una MV de servicio gratuito, tenemos para elegir los siguientes parámetros



Anexo A - Ilustración 29: Creacion de una Máquina Virtual

Sistemas Operativos disponibles

- Centos 6.5 HPC
- Centos 6.8 HPC
- Centos 7.1 HPC
- Centos 7.3 SRIOV
- Centos 7.3 HPC
- Centos 7.3
- Debian 8
- Debian 8 con backports kernel
- Debian 9 (Indica que es el 8, pero la versión 9 es Stretch, no Jessie)
- CoreOS Linux Alpha (Sistema enfocado a montar contenedores)
- CoreOS Linux Beta
- CoreOS Linux Stable
- Ubuntu 14 LTS
- Ubuntu 16 LTS
- Red Hat 6.7
- Red Hat 6.8
- Red Hat 6.9
- Red Hat 7.2
- Red Hat 7.3

- Red Hat 7.4

En las Máquinas con Windows se ofrece

- Windows Server 2008
- Windows Server 2012
- Windows Server 2012 R2 Datacenter
- Windows Server 2016 Datacenter
- Windows Server 2016 Datacenter Server Core

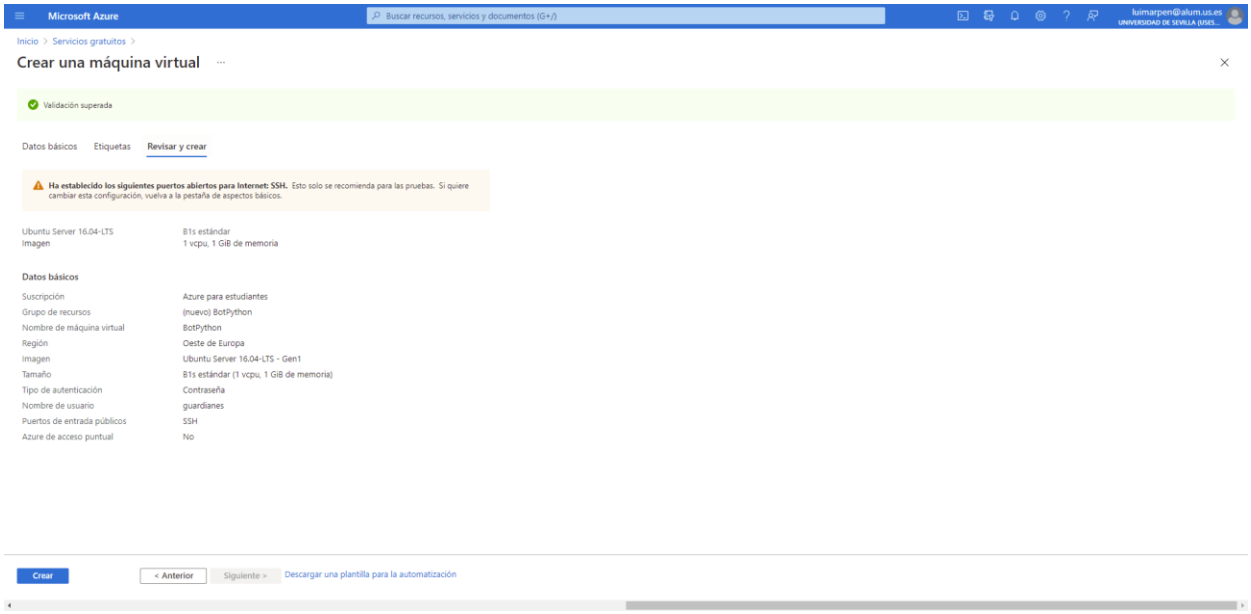
Hay que tener en cuenta lo que significa:

- SRIOV. Esto permite un acceso más directo al hardware de red para casos de necesidades mayores de red
- HPC. Permite escenarios de alto uso de recursos.

Luego añadiremos el tipo de autenticación y los puertos abiertos. Para poder acceder en remoto, es necesario utilizar un cliente SSH, como una consola desde Linux, o bien en Windows algún software como MobaXterm o Putty

Una vez terminado de configurar, podemos pulsar en Revisar y Crear para confirmar la creación de la MV.

Una vez creada se puede configurar operaciones de Apagado Automático (Muy útil si estamos utilizando una instancia de pago)



Anexo A - Ilustración 30: Datos de una Máquina Virtual

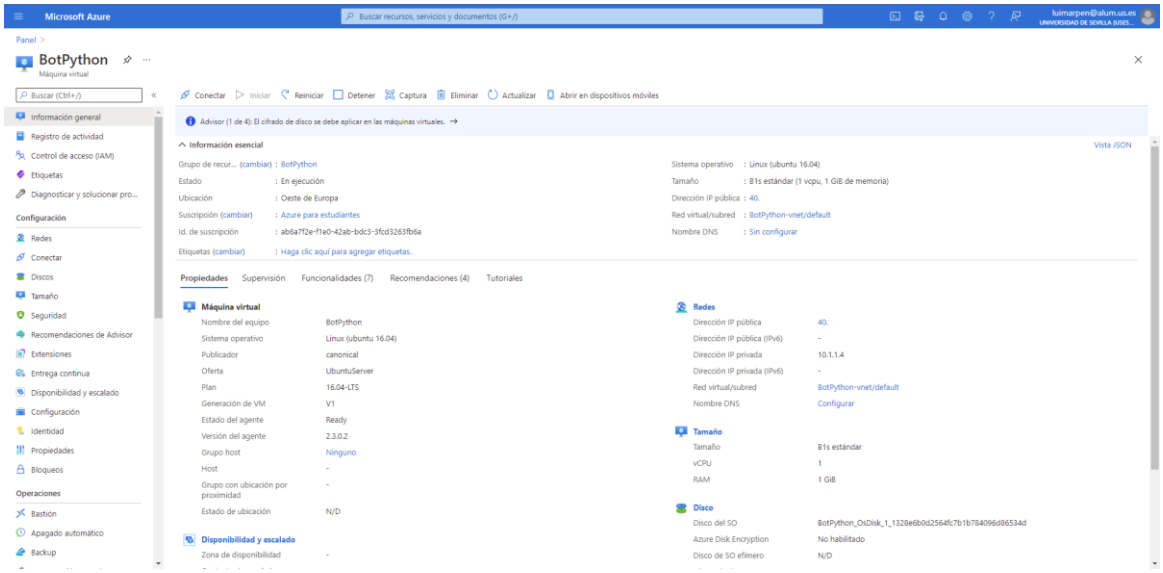
Características que tener en cuenta

- No se pueden escoger instancias de acceso puntual. Éstas son instancias que Microsoft Azure puede apagar cuando no están en uso o cuando el proveedor requiere usar capacidad para otras máquinas de otros clientes. A cambio se hace más barato su uso para máquinas que no requieren una gran cantidad de horas de procesamiento.

- Para las máquinas de pago sí se puede escoger instancias de acceso puntual. También permite la versión de pago usar otros sistemas operativos, añadiendo las versiones más modernas de Ubuntu tanto en Gen 1 o Gen 2.

- Las máquinas con Gen 2 permiten usar capacidades modernas de arranque y mejor uso de kernel y recursos. Es importante tener esto en cuenta si alguna de las versiones de SO anteriores no sirve para el despliegue del proyecto actual.

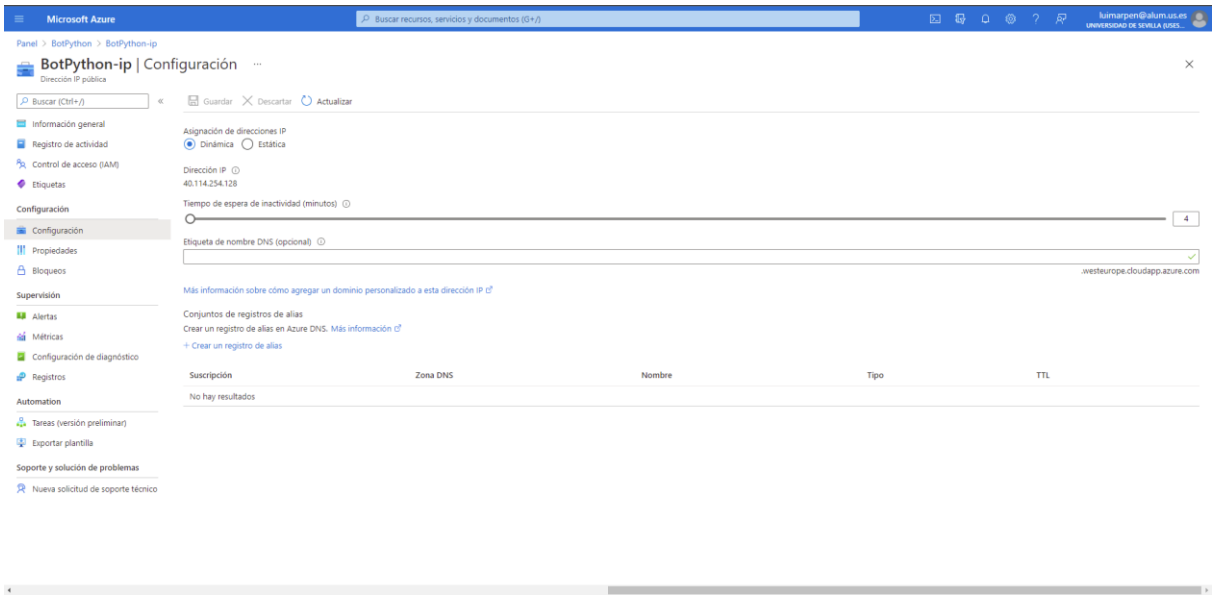
Una vez creada la máquina virtual podremos ir a la sección de Información General de la máquina virtual y ver las características de la máquina, así como la IP pública, necesaria en este caso para poder conectarse de forma remota.



Anexo A - Ilustración 31: Máquina virtual ya creada

Podemos pulsar en “Conectar” para ver las opciones de conexión. Si solo hemos permitido el acceso por SSH, solo podremos usar ese método, pero se pueden añadir más en la sección de Redes, abriendo puertos si es necesario.

Puede ser recomendable configurar un DNS para evitar los cambios de IP pública que pueden suceder al apagar la máquina. Para ello en la sección de Información General pulsaremos en Nombre DNS “Sin Configurar” y podremos configurar un nombre de dominio.



Anexo A - Ilustración 32: Configuración de red de Máquina Virtual

También se puede asignar una IP de forma estática en la misma sección.

Anexo B - Documentación de código a partir de comentarios

Para documentar el código se ofrecen funcionalidades en el entorno integrado de desarrollo, como pistas sobre los tipos de las variables, los parámetros de entrada y salida de una función o un método, entre otros. Estas funcionalidades se ofrecen de forma contextual al código, mientras se desarrolla.

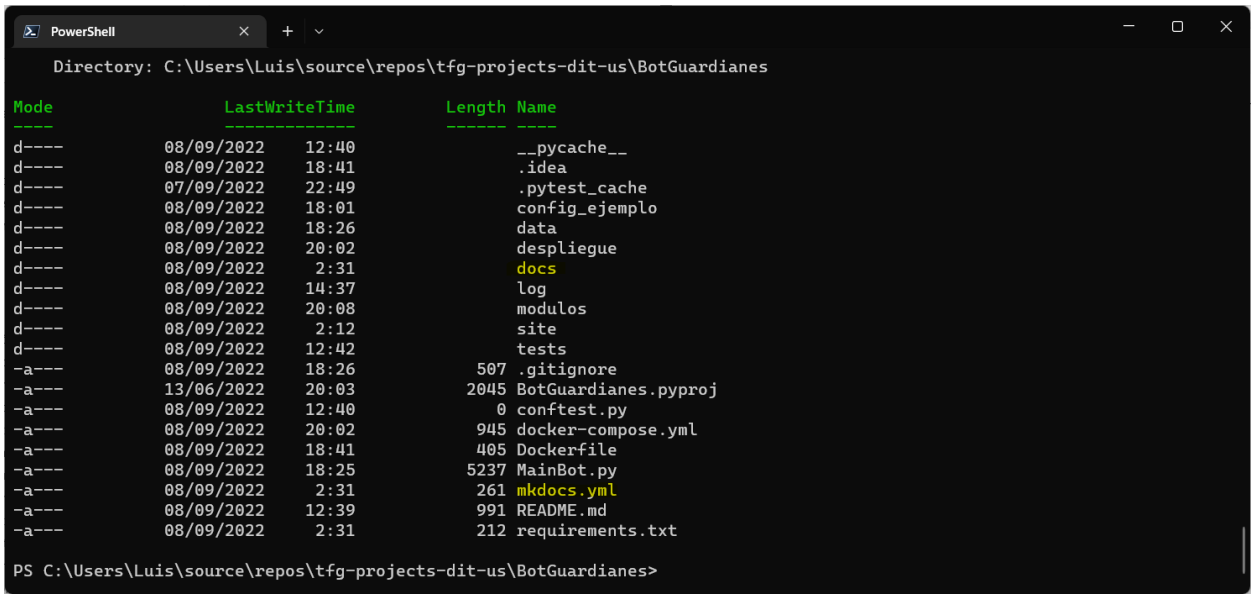
Para generar la documentación a partir del código, vamos a usar el módulo mkdocs.

Este módulo permite obtener un conjunto de páginas web que pueden ser publicadas.

En primer lugar, se abre una consola y navegamos a la carpeta donde se aloja el proyecto.

```
$ mkdocs new .
```

Esta sentencia genera una carpeta docs en la carpeta actual, además del fichero mkdocs.yml



Anexo B - Ilustración 33: Estructura del directorio del repositorio con MkDocs ejecutado

En el fichero mkdocs.yml incluimos datos como el tema que deseamos, el nombre de la web, y las páginas que queramos que tenga la documentación.

```
site_name: BotGuardianes

theme:
  name: "material"

plugins:
  - mkdocstrings

nav:
  - Principal: index.md
  - telegram_tools: telegram_tools.md
  - servicio_rest: servicio_rest.md
  - gestor_calendario: gestor_calendario.md
  - config: config.md
```

Anexo B - Código 4: Fichero mkdocs.yml

El plugin mkdocstrings se utiliza para que convierta las docstrings de comentarios de código en páginas web donde se desee incluir cada módulo.

Theme es el tema visual que utilizará la documentación. Usamos mkdocs-material para esta documentación

Nav es la estructura de navegación principal de la documentación. La forma de ordenarlo depende de cómo se pretenda crear la documentación.

Cada fichero indicado en nav refleja el nombre mostrado en la página y el fichero en /docs a la que se refiere dicha página

En el index.md está indicado cómo utilizar brevemente la web de documentación

En las demás solo se incluye una cadena que, tras procesarla, genera la documentación a partir de los comentarios.

Esto se hace incluyendo `:::paquete.modulo` al inicio de cada página. Por ejemplo, para el fichero servicio_rest.md, tiene como primera línea `:::modulos.servicio_rest`

Una vez creados estos ficheros, solo hemos de crear el sitio web con la instrucción

```
$ mkdocs build
```

Una vez creada la documentación, genera una carpeta “site” en el directorio, donde se aloja la web completa.

Mkdocs permite alojarla en los *sites* de Github directamente, siempre que el directorio actualmente esté funcionando con Git y configurado correctamente el repositorio remoto. Para ello, basta con usar la instrucción.

```
$ mkdocs gh-deploy
```

Al final de la ejecución, devuelve la URL donde se alojará la documentación.

Para activar los *sites* de Github, es necesario tener acceso de administrador al repositorio, y seguir ésta guía (https://docs.github.com/es/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site#publishing-from-a-branch)

7 REFERENCIAS

[1] M. Á. González-Alorda Cantero, *Servicio para gestión de actividades asistenciales complementarias,* Sevilla: Universidad de Sevilla, 2020.

[2] C. Cohen Calvo, *Gestión de calendarios para un servicio hospitalario,* Sevilla: Universidad de Sevilla, 2022.

[3] RFC Editor, «Calendaring Extensions to WebDAV (CalDAV),» 03 2007. [En línea]. Available: https://www.rfc-editor.org/rfc/rfc4791. [Último acceso: 19 08 2022].

[4] Python Software Foundation, «Acerca de Python,» [En línea]. Available: https://www.python.org/about/. [Último acceso: 19 07 2022].

[5] Telegram Group Inc, «Telegram Bot API,» [En línea]. Available: https://core.telegram.org/bots/api. [Último acceso: 19 07 2022].

[6] L. Toledo, H. Mahler, J. Höke, Pieter Schutz, P. Schutz y J. Bom, «python-telegram-bot,» [En línea]. Available: https://github.com/python-telegram-bot/python-telegram-bot. [Último acceso: 19 08 2022].

[7] K. Simonov, «PyYAML,» [En línea]. Available: https://github.com/yaml/pyyaml/. [Último acceso: 19 08 2022].

[8] S. Bishop, «pytz - World Timezone Definitions for Python,» [En línea]. Available: https://github.com/stub42/pytz. [Último acceso: 19 08 2022].

[9] A. Nyayachavadi, J. Chaar y K. F. Velkovski, «Arrow: Better dates & times for Python,» [En línea]. Available: https://github.com/arrow-py/arrow. [Último acceso: 19 08 2022].

[10] Python Software Foundation, «Requests: HTTP for Humans,» [En línea]. Available: https://github.com/psf/requests. [Último acceso: 19 08 2022].

[11] T. Brox y C. Robert, «Caldav client for Python,» [En línea]. Available: https://github.com/python-caldav/caldav. [Último acceso: 19 08 2022].

[12] Plone®, «Internet Calendaring and Scheduling (iCalendar) for Python,» [En línea]. Available: https://github.com/collective/icalendar. [Último acceso: 19 08 2022].

[13] RFC Editor, «Internet Calendaring and Scheduling Core Object Specification,» [En línea]. Available: https://www.rfc-editor.org/rfc/rfc5545. [Último acceso: 19 08 2022].

[14] T. Christie, «MkDocs,» [En línea]. Available: https://www.mkdocs.org/. [Último acceso: 01 09 2022].

[15] H. Krekel, «pytest,» [En línea]. Available: https://docs.pytest.org. [Último acceso: 01 09 2022].

[16] L. Torvalds y J. Hamano, «Git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 19 08 2022].

[17] R. Fielding, «REST API,» [En línea]. Available: https://restfulapi.net/. [Último acceso: 19 08 2022].

[18] S. Hykes, «Docker. ¿Qué es un contenedor?,» [En línea]. Available: https://www.docker.com/resources/what-container/. [Último acceso: 01 09 2022].

[19] A. McMillan, M. Delorme, R. Ostenson y L. Kraav, «DAViCal Home,» [En línea]. Available: https://www.davical.org/index.php. [Último acceso: 01 09 2022].

[20] B. Okken, «pytest_check,» [En línea]. Available: https://pypi.org/project/pytest-check/. [Último acceso: 01 09 2022].

[21] Google Inc., «Google Style Python Docstrings,» [En línea]. Available: https://google.github.io/styleguide/pyguide.html#s3.8-comments-and-docstrings. [Último acceso: 01 09 2022].

[22] University of Luxembourg, «Université du Luxembourg,» [En línea]. Available: https://wwwen.uni.lu/university/high_performance_computing. [Último acceso: 06 08 2022].

[23] VMware, «Virtualización de E/S de raíz única (SR-IOV),» [En línea]. Available: https://docs.vmware.com/es/VMware-vSphere/7.0/com.vmware.vsphere.networking.doc/GUID-CC021803-30EA-444D-BCBE-618E0D836B9F.html. [Último acceso: 06 08 2022].