UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aplicación de Gestión de Aulas mediante API RESTs de Microsoft

Presentado por Jorge Gómez Ortiz
en Universidad de Burgos — 19 de julio de 2020
Tutor: Jesús Maudes Raedo

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Jesús Maudes Raedo, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Jorge Gómez Ortiz, con DNI 71308418T, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Aplicación de Gestión de Aulas mediante API RESTs de Microsoft.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 19 de julio de 2020

Vº. Bº. del Tutor:

Jesús Maudes Raedo

Resumen

El proyecto consiste en facilitar la reserva de aulas de las universidades a los profesores que quieran reservarlas mediante el uso de las API REST de Microsoft.

Descriptores

Aplicación web, reserva de aulas, solapamiento de horas, Outlook, Flask, calendarios...

Abstract

The project is about to ease the reservation of university's rooms to the teachers who want to book them by using the Microsoft's API REST.

Keywords

Web application, classroom reservation, overlapping hours, Outlook, Flask.

Índice general

Índice de figuras

Introducción

La gestión de las reservas de las aulas siempre ha sido un problema en el que se ven envueltos los profesores que quieren reservar y los encargados de cada aula en cuestión. Esta aplicación web da solución a este problema, ya que permite la visualización de eventos y aulas a todos los usuarios y la posibilidad de realizar reservas de forma sencilla para usuarios que tengan cuenta de *Office 365* o Microsoft (gestionado por el login de Microsoft que utiliza *OAuth2* como estándar de autorización) y los permisos necesarios de las aulas sin que existan solapamientos entre dichas reservas. En este proyecto los calendarios, los eventos y las horas se gestionarán desde la API REST de Microsoft, en concreto utilizando el punto de enlace de *Microsoft Graph*.

Objetivos del proyecto

En esta sección se enumeran los objetivos que se tratan de abarcar en el proyecto.

Objetivos principales

- Desarrollar una aplicación web en la que los usuarios que dispongan de los permisos necesarios puedan reservas aulas.
- Mostrar al usuario que acceda a la aplicación web las franjas horarias reservadas y libres de cada aula.
- Permitir al administrador la creación de nuevos aulas, donde se podrán agregar eventos.
- Posibilidad para el administrador de modificar y borrar la información de las aulas.
- Filtrar las aulas según características para una mayor facilidad a la hora de reservar.
- Permitir el inicio y cierre de sesión mediante los servicios de autenticación de Microsoft.
- Avisar mediante un envío de email de la reserva realizada al profesor que la solicitó.

Objetivos técnicos

- Desarrollar una aplicación web cliente servidor mediante Flask.

- Utilizar la API REST de Microsoft para el inicio de sesión y el uso de los calendarios de cada aula.
- Utilizar el sistema de control de versiones de Git.
- Utilizar ZenHub dentro de GitHub para gestionar el proyecto.
- Desplegar la aplicación web y la base de datos en la nube.

Conceptos teóricos

Frameworks de desarrollo

Un framework es un entorno o marco de trabajo estandarizado, con una estructura conceptual y tecnológica que puede servir de base para la organización y desarrollo de software [18].
En este proyecto he utilizado como framework Flask, que se explicará en las herramientas usadas.

API REST

Üna API REST es cualquier interfaz entre sistemas que utilice directamente una llamada de HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc)". Esto permite que una aplicación puede interactuar con un recurso proporcionado en REST conociendo el identificador del recurso y la acción requerida para obtener la finalidad deseada [24].
En nuestro caso se ha utilizado el endpoint de Microsoft Graph para acceder a la API REST de Outlook, para ello hubo que realizar lo siguiente:
Primeramente se utiliza la API de Microsoft para para realizar el login en la aplicación mediante una cuenta de Outlook. Utiliza OAuth 2.0, que necesita el token recibido al registrar la aplicación en Azure Active Directory y nos devuelve un nuevo token. Una vez lo recibimos podemos hacer uso del endpoint de Microsoft Graph mediante este token de OAuth recibido.
Una vez se dispone de este acceso, se realizan llamadas mediante el punto de conexión Microsoft Graph: punto de conexión común para acceder a la API REST que permite el acceso y la modificación a todas las características de una cuenta Microsoft O365. Dentro de este punto de conexión se encuentran

las llamadas al resto de las APIs de Microsoft para lo que necesitemos (calendarios, usuarios, grupos de calendarios, correo).

APIs de validación

Permite implementar el inicio de sesión que utiliza una aplicación en tu propio proyecto, de forma que si existe esa cuenta permite el acceso a la dirección que se haya especificado. Para que este proceso funcione, Microsoft utiliza OAuth v2.0, que sigue un flujo de autorización y autenticación para acceder a la aplicación y redireccionar al usuario a la página indicada en Azure Active Directory. Este proceso devuelve un token de acceso a la aplicación que sirve para obtener acceso a los recursos protegidos mediante el punto de conexión de la API utilizada y otro token para refrescar el código de acceso anterior en caso de que caduque si el inicio de sesión es correcto. Un token es un código de letras y números único que se necesita para realizar el resto de llamadas a la API en la que nos hemos logeado para obtener información sobre la sesión que hemos iniciado.

APIs de calendarios

La API de un calendario nos permite guardar eventos dentro de los calendarios que nos proporciona, borrar, modificar o compartir estos eventos y calendarios. Mediante llamadas a esta API se pueden acceder a los calendarios, recuperar los eventos y realizar las distintas modificaciones sobre ellos. Esta API que hemos usado pertenece a la API REST de Outlook y accedemos a ella mediante el punto de acceso de Microsoft Graph como se ha explicado anteriormente.

APIs de conectividad de BD

Es una capa intermedia de acceso a las bases de datos que hace posible las acciones *CRUD* a cualquier dato desde la aplicación que desarrollamos [17]. En en el presente trabajo utilizamos *pyodbc*: Es un módulo de Python de código abierto que simplifica el acceso a las bases de datos ODBC. Usado para realizar la conexión entre la base de datos y la aplicación web desarrollada, alojadas ambas en Azure.

Técnicas y herramientas

4.1. Técnicas

Scrum

La metodología *Scrum* es una metodología de desarrollo ágil, en la que se aplican buenas prácticas y procesos con la finalidad de obtener la mejor calidad posible en el producto final buscado [26].
Esta metodología está pensada para un equipo de trabajo, con el fin de organizarse y trabajar más eficientemente obteniendo los mejores resultados posibles. En el presente trabajo se trató de simular una metodología *Scrum* ya que el trabajo lo realiza un único alumno, pero la intención es dividir dicho trabajo en *sprints* que contendrán dentro las tareas a realizar.
Esta metodología se basa en crear *sprints*, dentro de los que se crearán las tareas a realizar (*issues*). Para establecer estas tareas dentro de cada sprint, se realiza una planificación, en la que se deja claro los objetivos a conseguir y el tiempo para cada uno de ellos. Se realizan revisiones para controlar los avances de los sprint planificados y llevar un control sobre el proyecto.

4.2. Herramientas

GitHub

Para el control de versiones de este proyecto he utilizado *GitHub*, que es un repositorio en línea que emplea *Git*. Así podemos guardar los cambios que vamos realizando en el proyecto y controlar que se cumplan las tareas en el tiempo indicado. Para llevar un mejor control de las tareas activas y

lo pendiente se utilizó ZenHub, que facilita un panel en el que se cambian los estados de dichas tareas.

GitTortoise

Es el cliente de control de revisiones que he utilizado para subir los cambios y tener actualizado el GitHub. Permite comparar versiones y solucionar conflictos entre ellas, facilita la subida de código.[25]

Visual Studio Code

Editor y compilador de código fuente. Incluye el soporte que ofrece para la depuración y para el control integrado de *Azure Web Services* el cual ha sido utilizado en el presente trabajo.

JavaScript

Es un lenguaje de programación ligero e interpretado, orientado a objetos pero basado en prototipos. En mi caso utilizado como lenguaje de scripting para páginas web. Se ha utilizado desde el lado del cliente proporcionando mejoras de las interfaz y de la comunicación con el servidor [20].

LaTeX

Como herramienta para realizar la documentación se ha escogido *LaTeX*, está diseñado para crear documentos con una alta calidad tipográfica. Como editor de *LaTeX* he utilizado Overleaf

Azure

Es un servicio de computación en la nube, de Microsoft, diseñado para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Proporciona muchas herramientas para facilitar el manejo de estas aplicaciones o servicios. En el caso de este trabajo se ha utilizado para desplegar la aplicación web y la base de datos. Ambas pueden ser gestionadas desde extensiones del Visual Studio Code, el programa que utilizo para el desarrollo del proyecto. [22].
También ha sido necesario utilizar el servicio de administración de identidades de Azure, conocido como Azure Active Directory, para registrar la aplicación y poder proporcionar un inicio de sesión único a ésta.

Outlook API REST (Microsoft Graph)

Para realizar este proyecto se ha utilizado la API REST de Outlook para manejar los calendarios, que equivalen a las aulas que se proporcionan al usuario para reservar. Para tener acceso a las herramientas de esta API REST, se usó el punto de conexión Microsoft Graph ya que ofrece más servicios y características siguiendo la recomendación de Microsoft (https://docs.microsoft.com/es-ES/outlook/rest/compare-graph [5]). En concreto para este proyecto he utilizado distintas funciones de esta API REST:

- Calendarios: Para realizar modificaciones u obtener información de los calendarios del usuario.
- Correo: Para enviar los mensajes automáticos.
- Grupos de calendarios: Para obtener los grupos de calendarios (equivalen a los edificios) y la información que contiene cada uno de ellos.

Una vez tenemos la cuenta iniciada recibimos los tokens de autenticación y de refresco, como ya vimos anteriormente, estos se utilizan para validar la sesión si se caduca el token de acceso o recoger uno nuevo al iniciar una sesión distinta, para ello se envía a la dirección proporcionada por Microsoft al registrar la app.
Dentro de este punto de conexión a la API REST hemos utilizado varias llamadas distintas, las más destacables y usadas en el proyecto son relacionadas a las llamadas de los calendarios, estas son las siguientes:

- Obtener el id de los calendarios del usuario que esté logeado o del grupo de calendario especificado.
- Añadir eventos nuevos al calendario especificado mediante el id obtenido en el punto anterior.
- Obtener los grupos de calendarios (que en este proyecto los conocemos como edificios).
- Obtener los id de los eventos de cada calendario para poder modificarlos o borrarlos.

Pylint

Pylint es una herramienta de análisis de código estático de Python que utilizo en Visual Studio Code, busca errores de programación, ayuda a

aplicar un estándar de codificación y ofrece sugerencias de refactorización simples. Es un software altamente configurable y gratuito.[3]

SQL Server

Es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft, utilizado por Azure para gestionar los servicios de bases de datos en la nube. El entorno de trabajo que se ha utilizado ha sido la propia página de Azure, que permite la creación de tablas y ejecución de consultas, ya que primeramente fue creado en MySQL y solamente había que trasladarlo.[23]

JWT

JWT (JSON Web Token) es un estándar en el que se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario. Se trata de una cadena de texto que tiene tres partes codificadas en Base64, cada una de ellas separadas por un punto. En este proyecto, necesitamos una función que pueda decodificar el JWT del id recibido al iniciar sesión, para obtener la información necesaria como token de acceso, token para refrescarlo o email de la sesión.[13]

Herramientas web

Flask

Flask es un 'micro' framework en lenguaje Python que permite y facilita la creación de aplicaciones web con un comienzo sencillo y una escalabilidad muy alta. Utiliza la especificación Werkzeug y Jinja[21] como motor en los templates.
Que sea un framework de tipo 'micro' significa que al instalar Flask tenemos las herramientas necesarias para crear una aplicación web funcional, donde se mantiene el núcleo simple pero extensible, a diferencia de frameworks full stack que proporcionan herramientas y módulos internos para soportar aplicaciones más grandes [14].

Jinja2

Jinja2 es un motor de plantillas web para aplicaciones desarrolladas en *Python*. Permite que *Flask* pueda hacer uso de los contenidos de las plantillas *HTML* [21].

Bootstrap

Es una biblioteca de código abierto que proporciona estilos, plantillas y diseños ya creados para usar directamente en nuestras aplicaciones web [19].

Werkzeug

Werkzeug es una biblioteca de aplicaciones web de WSGI (interfaz de puerta de enlace del servidor web), se utiliza internamente en Flask y entre otras muchas, realiza las siguientes funciones:

- Depurador para inspeccionar rastros de la pila y el código fuente.
- Objeto de solicitud para interactuar con encabezados, argumentos de consulta, datos de formulario, archivos y cookies.
- Objeto de respuesta en la transmisión de datos.
- Enrutamiento de URL.
- Utilidades HTTP.

Aspectos relevantes del desarrollo del proyecto

En este apartado mostraré los aspectos más interesantes a la hora de desarrollar el proyecto, esto hace referencia a problemas surgidos, los caminos tomados para avanzar, el desarrollo de la funcionalidad.

Preparación

Como ya se ha comentado en la sección anterior, no se disponían de los conocimientos suficientes para el desarrollo de esta aplicación web, sobretodo tratando con lenguaje Python, exactamente con el *framework Flask*. Sobre *HTML* y *JavaScript* ya estaba más relacionado.También se tuvo que tratar con Microsoft Graph, desde dónde se gestionarían los eventos y calendarios. Para aprender sobre estos temas antes de comenzar seguí los siguientes tutoriales:

- The Flask Mega Tutorial[9]
- J2logo[10]

Cuanto más avanzaba el proyecto, más herramientas y distintas técnicas hacían falta, por lo que se tuvo que usar mucha documentación para comprender los funcionamientos de estas, las más consultadas han sido:

- Microsoft Graph[5]
- Documentación Flask[15]

- Bootstrap[2]

Puesta en marcha del entorno de producción y pruebas - Resumen

Para comenzar el proyecto en Flask se necesitó tener instalado Visual Studio Code desde donde programaríamos nuestra web app, para ello hay que realizar unas instalaciones de requisitos que iban a ser necesarias para el desarrollo del proyecto desde el VSCode:

1. Creación de un entorno virtual, para aislar las dependencias que requiere el proyecto de las ya existentes.

2. Pylint como verificador del código que escribimos.

3. Git Tortoise para facilitar las subidas a GitHub.

4. Instalar extensiones de VSCode.

5. Registrar la aplicación en Azure Active Directory.

Para comenzar el proyecto hay que hacer un primer registro de la aplicación en Azure Active Directory (explicado en el manual del programador). Esto permite acceder a la sesión de Outlook y poder tener un token de acceso para recoger información sobre esta sesión y comenzar a trabajar con los calendarios de cada uno.

Otro aspecto a remarcar es la conexión con la base de datos, al subir todo a los servicios de Azure, la conexión ya no se puede realizar como al principio, que se hacía con la base de datos local de MySQL, ahora se realiza una conexión de pyodbc especificando el servidor, la base de datos, el usuario, contraseña y driver a utilizar desde Azure. Esto no permitirá que sea ejecutado en cualquier servidor local como se verá más adelante en la explicación del despliegue en el manual del programador. Se tuvo que modificar las consultas que realizaba ya que se tuvo que crear una nueva conexión distinta a la ya existente y el lenguaje de consulta cambia de MySQL al utilizarlo en SQL Database proporcionado por Azure.

Adaptación y dificultades en el proceso de desarrollo

En esta sección nos centraremos exclusivamente en los problemas que fueron surgiendo durante el desarrollo y como los tratamos.
Más adelante vimos que nos resultaría más cómodo trabajar mediante el uso de calendarios compartidos ya que todos serían gestionados desde una cuenta administrador en Outlook y estos pueden ser compartidos con los permisos deseados a los propietarios de las aulas, que serán los encargados de realizar reservas sobre dichos calendarios. De esta manera generamos calendarios globales, que equivalen a aulas, y están disponibles para todos los usuarios con los que se compartan. El problema de esto se ve a continuación, ya que compartir estos calendarios tiene que ser una función a realizar por el administrador manualmente.

Como se acaba de mencionar, el error que no se ha podido solventar es automatizar el compartir calendarios desde la aplicación web, ya que no proporciona esta funcionalidad la API REST de Microsoft. Se estuvo buscando en la documentación de la que obtenemos todas las funcionalidades de dicha API REST y lo único que encontramos, fue una funcionalidad en fase Beta aún parte de ella aún y la funcionalidad que nos ofrece es la de gestionar los permisos de un calendario compartido con algún usuario, como vemos en la cita textual de esta página *'Antes de que se pueda aplicar el uso compartido del calendario o la delegación, el propietario envía una invitación a una persona con la que compartir o delegado, y dicha persona o delegado acepta la invitación'*[7]. Investigando y dando vueltas a este problema, se llegó a la conclusión de que la única forma de compartirlos es manualmente desde la cuenta del administrador. En el manual del programador se explica como compartir los calendarios a los propietarios.

Un aspecto relevante del proyecto fue la elección del punto de enlace de Microsoft Graph en vez del *endpoint* de la API de Outlook, ya que ofrece más servicios y características, como por ejemplo los más usados en la aplicación que son usuarios, calendarios o correos, permitiendo que desde el mismo punto de conexión se trabaje con todo esto. Elección tomada según la documentación de Microsoft [4]. Primeramente se estuvo trabajando con el extremo de la API de Outlook (https://outlook.office.com/api) en la v2.0, ya que parecía más nuevo y posiblemente mejor, pero fueron apareciendo errores a la hora de utilizarlo y falta de características. Lo que provocó invertir varios días en una línea infructuosa que retrasó el avance del sprint, teniendo que buscar una solución, y finalmente encontrando que cambiar los

puntos de conexión a Microsoft Graph era la mejor, esto significó cambiar la dirección URL a la que se realizan las llamadas de las APIs, adaptando los correspondientes parámetros según el tipo de llamada.

Durante el desarrollo fueron surgiendo problemas y diferencias en el uso de las API con el tutorial de Microsoft, ya que está diseñado para Django y no para Flask, aquí se tuvo que decidir si cambiar el comienzo del proyecto y empezar de nuevo con Django para tener este aspecto solucionado o buscar la alternativa en el lenguaje de Flask, ya que la imposición del tutor para el proyecto fue utilizar este framework en el desarrollo, por lo que la solución tomada fue adaptar el código, esto implicó cambios en el código disponible en la documentación existente en [1] y obligó a buscar una adaptación de distintos características que tenían que ser usadas. Por ejemplo, para obtener el JWT (JSON Web Token), no sirve la función proporcionada ya que se necesitan diferentes argumentos en la función, mediante un identificador de token proporcionado por una llamada a la API REST en este caso.

Despliegue en Azure

En este apartado se aborda otro aspecto importante, que fue la decisión de subir el proyecto a Azure, ya que se había registrado la aplicación en Azure Active Directory y facilitaba una base de datos y permitía alojar la aplicación web sin coste.
En cuanto al trabajo con Azure, comentar que para realizar la subida se tuvo que reestructurar el proyecto según lo que Microsoft establece al trabajar en un proyecto que utiliza VSCode y Flask [12] [16]. Esto conllevó errores en las partes del código que se importaban al cambiar la carpeta raíz desde VSCode. Siguiendo las mismas especificaciones también generé el fichero *launch* para poder ejecutar desde local mediante el debug.
Una vez que el proyecto estaba estructurado se siguieron los pasos del tutorial que proporciona Miscrosoft para realizar la subida de la aplicación web a los servidores de Azure. Despliegue en Azure[6].

Por último quiero comentar el problema que se tuvo con la suscripción de Azure. En primera instancia se disponía de una suscripción gratuita de estudiante, desde la cuál no se consiguió mantener abiertos los servicios desplegados las 24 horas del días, para poder ejecutarlo hay que iniciar el app service y la base de datos previamente. Pero al final del proyecto se agotó la prueba gratuita de estudiante desde la que ejecutaba el app service

y la base de datos, por lo que se tuvo que contactar con el soporte de Azure y pedir un aumento del tipo de suscripción, ya que no se podía realizar esto manualmente, de modo que convirtieron la suscripción de estudiante en una suscripción por pago, lo que hace posible tenerlo subido el día de la defensa sin tener que iniciarlo justo antes manualmente.

Aportaciones

En esta sección se van a tratar las aportaciones a nivel personal más interesantes del proyecto.
En primera instancia, me gustaría comentar las consultas que se realizan a la base de datos, tanto a la hora de reservar un aula para comprobar si está libre como a la hora de comprobar los eventos de un aula en concreto, ya que estas acciones no las realiza la API REST. Esto se realiza mediante unas SELECT que comprueban todos los posibles solapamientos dividiendo las fechas en el día y las horas para poder comparar con más precisión. A la hora de consultar las reservas de las aulas, dependiendo de los filtros que se establezcan se realizará una consulta u otra, lo mismo para a la hora de realizar la reserva.
También hemos abordado el tema de las reservas en un entorno multiusuario, de forma que al intentar realizar una reserva, se ejecuta una primera consulta sobre el aula en el que se va a realizar la reserva, de tipo *HOLDLOCK* provocando el bloqueo de esta para cualquier otro usuario hasta que termine la transacción.

Trabajos relacionados

Esta aplicación web tiene similitudes con todas las aplicaciones de reserva de espacios, pero en lo que se distingue esta aplicación principalmente es en el uso de Outlook tanto para la validación del usuario como para la gestión de los calendarios usando Outlook calendar.
En cuanto a funcionalidades, mirando aplicaciones similares de reservas, la principal diferencia a destacar es que ésta está realizada para la universidad o entidades similares, ya que se tiene en cuenta distintos edificios en los que se manejan distintas aulas y dichas aulas tienen características y se proporcionan filtros concretos de aulas de universidad.
Al hacer una búsqueda de aplicaciones que guardan cierta similitud con la presente, se han encontrado las siguientes:

- *BookedApp*
  Es una aplicación que se dedica a la reserva de sitios individuales, no de aulas, sino de bibliotecas, a diferencia de la aplicación desarrollada en el presente trabajo que gestiona más de un espacio pero no por sitios individualmente, sino reservas de aulas completas. [8]

- *Booked Scheduler – Reserva de aulas*Enlace[11]
  Es una aplicación más desarrollada que permite reservas de aulas de forma similar a la del presente trabajo, pero no está sincronizado con Outlook y todo se gestiona a través de una base de datos, sin embargo como ventaja suya podemos apreciar los eventos representados en instancias de calendarios.

Conclusiones y Líneas de trabajo futuras

Este proyecto me pareció útil e interesante, ya que se centra en el desarrollo web de una aplicación que serviría para facilitar la gestión de las reservas de las aulas. Este es un problema que se da en muchos lugares a la hora de reservar y la idea es facilitar la tarea de las reservas y que no ocurran solapamientos entre dichas reservas.
Una primera conclusión que me gustaría dejar es relacionada al trabajo con Azure, ya que el despliegue se realizó ahí como se ha comentado a lo largo de la documentación. El problema que se encontró al final, respecto al fin de licencia de estudiante y tener que ampliarlo a una suscripción por pago. En este proyecto trabajando con una base de datos y un app service, los servicios hay que arrancarlos antes de poder ejecutarlo en la nube, se puede establecer que no haya pausas, surgiendo nuestro problema, donde se gastó el dinero al facturar como ejecutando continuamente la base de datos. Por lo que sugiero una diferente plataforma o alguna alternativa como quizás no ejecutar las acciones que trabajan con la base de datos en cada ejecución, ya que desde local también trabajaba con esta base de datos subida. Finalmente y tras conversar con el servicio técnico de Azure, se llegó a la conclusión de que para la presentación del proyecto sería mejor un cambio en la configuración de la base de datos , estableciendo un uso básico para que esté abierta continuamente y se pueda probar en cualquier momento.
Otro impedimento que tuve al utilizar Azure fue la imposición de utilizar SQL Server en vez de MySQL como estaba utilizando al comienzo en local, esto se debe a que en la suscripción de Azure se permite crear gratuitamente bases de datos pero de SQL Server solamente.

Como conclusiones del desarrollo del proyecto, centrándome en el trabajo que se ha realizado con la API de Microsoft, es importante destacar que Microsoft está actualmente implantada en la mayoría de instituciones educativas, como bien podemos obtener las licencias de sus aplicaciones, por lo que a la hora de querer implantarlo dentro de la universidad no se tendrán problemas de licencias y todo estará enlazado más fácilmente.
Sobre esta API REST en concreto cabe destacar su ayuda para alcanzar los resultados, pero dicha API aún tiene funcionalidades en desarrollo y en versión Beta, por lo que se puede mejorar en este aspecto, como por ejemplo introduciendo la posibilidad de compartir un calendario con otra persona mediante una llamada a la API, esta funcionalidad apareció en la documentación con el proyecto muy avanzado y para poder hacer uso de ella requiere de invitación manual, ya que se centra en administrar el uso compartido y la delegación de permisos sobre los calendarios ya compartidos. Este aspecto queda como una mejora futura de la aplicación, de forma que desde la propia aplicación se pueda compartir el aula con el propietario que se le asigne, de momento cualquier cambio sobre el propietario del aula hay que asignárselo manualmente desde la cuenta del administrador, que controla todas las aulas, compartiendo el calendario con la persona que sea propietario.
Otro punto que ralentizó el avance del proyecto fue el problema de los solapamientos, ya que la API puede devolver los eventos existentes, pero para controlar todas las horas que se puedan solapar, no sólo que se muestre como solapamiento la misma hora hubo que recurrir a consultas de la base de datos. Los calendarios de Outlook permiten los solapamientos y no disponen de una función que bloquee esto.
En cuanto al entorno de trabajo ningún problema al trabajar con Visual Studio Code, que dispone de una documentación bien detallada y de muchas extensiones que agilizan el trabajo.
Otra posible mejora futura sería una integración con ubuVirtual, como acabo de mencionar mediante el uso de Microsoft ya existente, para que las reservas aparezcan en el calendario de exámenes si fueran exámenes o como notificaciones de eventos especiales en caso de charlas o talleres.

Bibliografía

[1] Python en microsoft graph. https://docs.microsoft.com/es-es/graph/tutorials/python.

[2] Bootstrap team. https://getbootstrap.com/docs/4.5/getting-started/introduction/, 2018 (accedido 29/06/2020).

[3] Python Code Quality Authority. Pylint. https://pypi.org/project/pylint/, 2020 (accedido 26/06/2020).

[4] Microsoft Corp. Microsoft graph comparación api outlook. https://docs.microsoft.com/es-ES/outlook/rest/compare-graph, 2017 (accedido 02/07/2020).

[5] Microsoft Corp. Comparar los extremos de microsoft graph y la api de rest de outlook. https://docs.microsoft.com/es-ES/outlook/rest/compare-graph, 2017 (accedido 28/06/2020).

[6] Microsoft Corp. Deploy azure app service en vscode. https://docs.microsoft.com/es-es/azure/developer/python/tutorial-deploy-app-service-on-linux-01, 2019 (accedido 26/06/2020).

[7] Microsoft Corp. Compartir o delegar un calendario en outlook. https://docs.microsoft.com/es-es/graph/outlook-share-or-delegate-calendar?tabs=http, 2020 (accedido 26/06/2020).

[8] Marina Calderón de la Barca Signès Mario García Suárez Manuel Valcárcel López; Elena Cuadrado del Arco (dir.) Elena Fernández Blanco (dir.) Francisco Micó Juan, Gonzalo del Palacio Jiménez. https://www.20minutos.es/noticia/3091727/0/alumnos-upsa-

crean-app-para-reservar-aulas-ver-ocupacion-biblioteca-tiempo-real/, 2017 (accedido 08/07/2020).

[9] Miguel Grinberg. Tutorial flask. https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world, 2017 (accedido 20/06/2020).

[10] José Lozano Gómez. Tutorial flask en español. https://j2logo.com/tutorial-flask-espanol/, 2018 (accedido 20/06/2020).

[11] Valentín Gómez. App booked scheduler. https://valentingom.wordpress.com/2016/07/10/booked-scheduler-reservas/, 2016 (accedido 26/06/2020).

[12] kraigb. Python sample. https://github.com/microsoft/python-sample-vscode-flask-tutorial, 2018 (accedido 26/06/2020).

[13] Luis Miguel López Magaña. Json web token. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/, 2020 (accedido 28/06/2020).

[14] José Domingo Muñoz. Qué es flask. https://openwebinars.net/blog/que-es-flask/, 2017 (accedido 26/06/2020).

[15] Pallets. Flask. https://flask.palletsprojects.com/en/1.1.x/, 2010 (accedido 20/06/2020).

[16] VSCode. Flask tutorial vscode. https://code.visualstudio.com/docs/python/tutorial-flask, 2019 (accedido 26/06/2020).

[17] Wikipedia. Odbc — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Open_Database_Connectivity, 2019 (accedido 28/06/2020).

[18] Wikipedia. Framework — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Framework, 2020.

[19] Wikipedia. Bootstrap (framework) — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Bootstrap_(framework)&oldid=125923117",note="[Internet;descargado23-junio-2020], 2020 (accedido 23/06/2020).

[20] Wikipedia. Javascript — wikipedia, la enciclopedia libre, 2020 (accedido 23/06/2020).

[21] Wikipedia. Jinja — wikipedia, la enciclopedia libre. https://en.wikipedia.org/wiki/Jinja_(template_engine), 2020 (accedido 23/06/2020).

[22] Wikipedia. Microsoft azure — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Microsoft_Azure&oldid=126900073",note="[Internet;descargado23-junio-2020], 2020 (accedido 23/06/2020).

[23] Wikipedia. Microsoft sql server — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Microsoft_SQL_Server&oldid=125873041",note="[Internet;descargado28-junio-2020], 2020 (accedido 28/06/2020).

[24] Wikipedia. Rest wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional, 2020 (accedido 28/06/2020).

[25] Wikipedia contributors. Tortoisegit — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=TortoiseGit&oldid=956271231, 2020 (accedido 20/06/2020). [Online; accessed 23-June-2020].

[26] Xavier Quesada Allue Jordi Salvat i Alabart Jesús Iglesias Gustavo Veliz Bernaola Martín Pérez Xavier Albaladejo, José Ramón Díaz. Metodología scrum. https://proyectosagiles.org/que-es-scrum/.