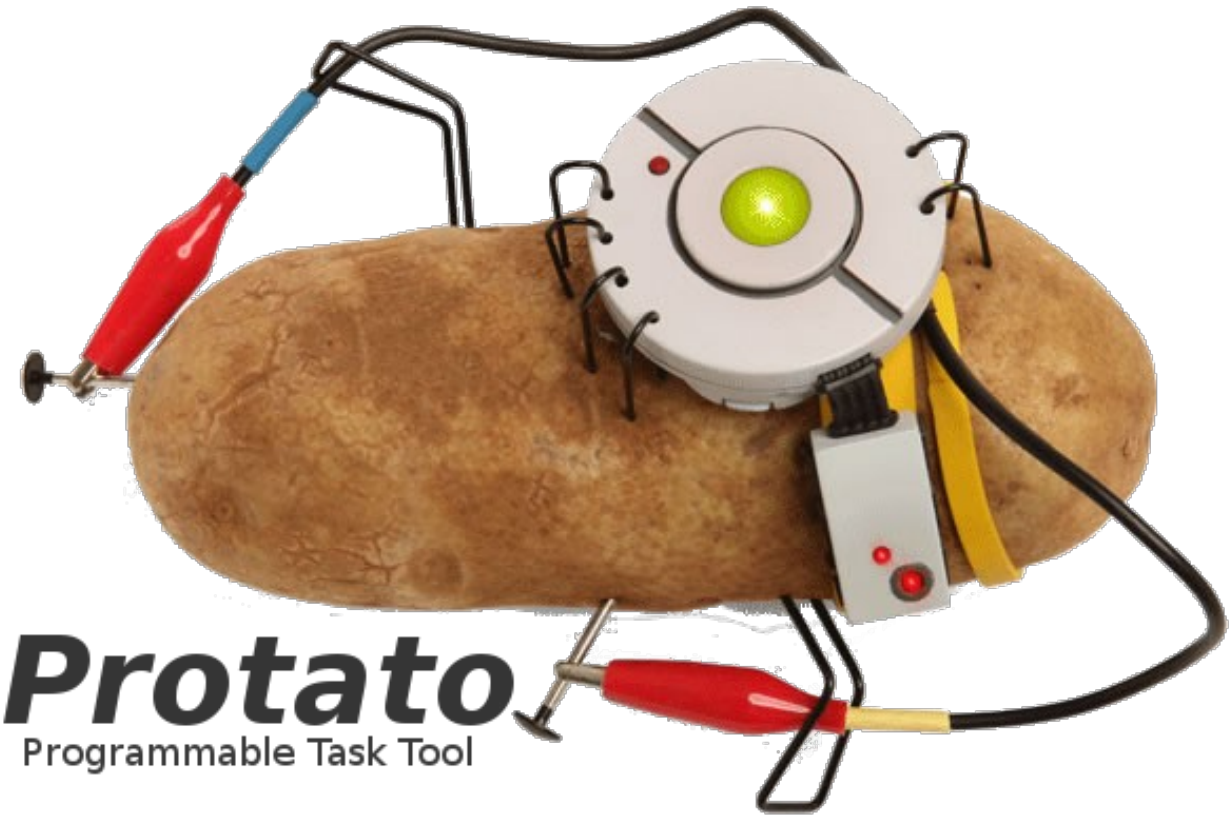


Tutores

Muñoz Rodríguez, José Domingo
Molina Coballes, Alberto

Centro

I.E.S Gonzalo Nazareno

Alumnos

Arias Gil, Jesús
Beleño Rodríguez, Álvaro
Hijano Martel, Carlos
Raya Bernal, Luis Manuel
Vela Fernández, José Manuel

Área de Participación

Informática de sistemas y de software

Primera Edición del Grow-lab ec2ce

ÍNDICE

1. Resumen Ejecutivo 3
2. Objetivo 3
3. Desarrollo 4
    3.1 Equipo Desarrollo 5
    3.2 Equipo BBDD 8
    3.3 Equipo Web 9
    3.4 Despliegue 13
4. Colaboración 14
5. Resultados 15
6. Conclusiones 15

1.- Resumen Ejecutivo

Los alumnos involucrados en el concurso **Grow-Lab**, organizado por la empresa **ec2ce**, decidimos que queríamos hacer un proyecto funcional y a ser posible que solucionase o paliara algún problema cotidiano de la mayoría.

Pensamos que con la vida ajetreada de hoy en día seria interesante conseguir aprovechar mejor un recurso tan valioso como es el tiempo. Y nos dimos cuenta que la mayoría de las tareas rutinarias son flexibles, en un rango horario. Por ejemplo, podemos ir a comprar durante toda la tarde aunque no nos lleve más de un par de horas. Y eso nos permite jugar con las tareas para poder organizarnos y aprovechar de forma más eficaz el tiempo.

Como alumnos, de Administración de Sistemas Informáticos, vimos una gran oportunidad de enfocar el proyecto en un ámbito cercano aunque diferente al nuestro, como desarrolladores de aplicaciones, programando una aplicación web. Capaz de organizar nuestro día a día y poder aprovechar al máximo nuestro tiempo de una forma eficiente y ser conscientes de todas las experiencias y proyectos personales que podemos hacer en esta etapa tan importante de nuestras vidas.

Y así es como surgió este emocionante y novedoso proyecto **Protato** *(Programmable Task Tool)*

El presente proyecto ha sido desarrollado y documentado a su vez en un repositorio de Github (https://github.com/iesgn/growlab18/) para su uso libre.

2.- Objetivos

El objetivo principal de **Protato** es poder organizar una multitud de tareas de forma fácil y rápida. Para ello decidimos idear una aplicación capaz de recoger y ordenar las diferentes tareas de los usuarios. Lo que denominamos un Organizador de Tareas Dinámico. Con el objetivo principal en mente pensamos también en todas las características que nos gustaría tener en la aplicación:

- **Interfaz amena y sencilla:** Con un aspecto simple pero robusto con (temática/interfaz) de calendario legible y comprensible a simple vista.

- **Diferentes tipos de tareas:** Para hacer de la agenda algo más que lo convencional, tenemos diferentes tareas como:
  - Rutinas diarias
  - Rutinas semanales
  - Rutinas mensuales
  - Rutinas anuales
  - Eventos fijos
  - Eventos flexibles

- **Sincronizar agendas:** Posibilidad de organizar eventos entre varios usuarios, como reuniones de trabajo, quedadas con amigos…

- **Multiplataforma:**
  - Ordenador: Mediante una pagina web.
  - Aplicación Móvil: Mediante una API Restful (Representational state transfer) que permitiera el desarrollo de una aplicación móvil que obtuviera la información de ella

Por tanto las funcionalidades que debemos desarrollar en la aplicación para cumplir con las características deberían ser:

- La aplicación sea multiusuario, por lo tanto dar la posibilidad de registro y autenticación de usuarios en nuestra aplicación. Cada usuario tendrá su calendario.

- Cada usuario gestiona su programación de eventos flexibles (rutinas diarias, semanales, fijas, ...), dando de alta, modificando o borrando los eventos en su programación.

- Cada vez que el usuario visualiza su calendario se calcula a partir de la programación de los eventos, las tareas a realizar, repartiéndolas de una manera "inteligente".

- Si durante el reparto de eventos en un determinado día, hay eventos programados que no se pueden repartir habrá que indicarlo al usuario.

- Los eventos repartidos de días pasados se deben guardar para ser consultados por el usuario en un momento determinado, el reparto de eventos se hará del día actual y de fechas futuras.

- Se podrá compartir un evento con uno o varios usuarios, por lo que dicho evento se repartirá en la agenda de cada uno de ellos en una misma hora.

3.- Desarrollo

Para conseguir el objetivo principal, de una aplicación web multiusuario capaz de gestionar las tareas, empezamos pensando cual iba a ser la lógica y los componentes de nuestra aplicación. Tras mucho debatir asentamos las bases de nuestra aplicación, definimos tres bloques fundamentales y nos dividimos en tres equipos.

3.1 Equipo de Desarrollo (algoritmo de ordenación):

Encargado de trabajar en el algoritmo que seria el motor de toda la aplicación. Capaz de diferenciar cada tipo de evento y ordenarlo en función de sus características. Aquí se encuentra el grueso del proyecto y debíamos definir muy bien la lógica del algoritmo para que fuera eficiente y flexible para futuras versiones.

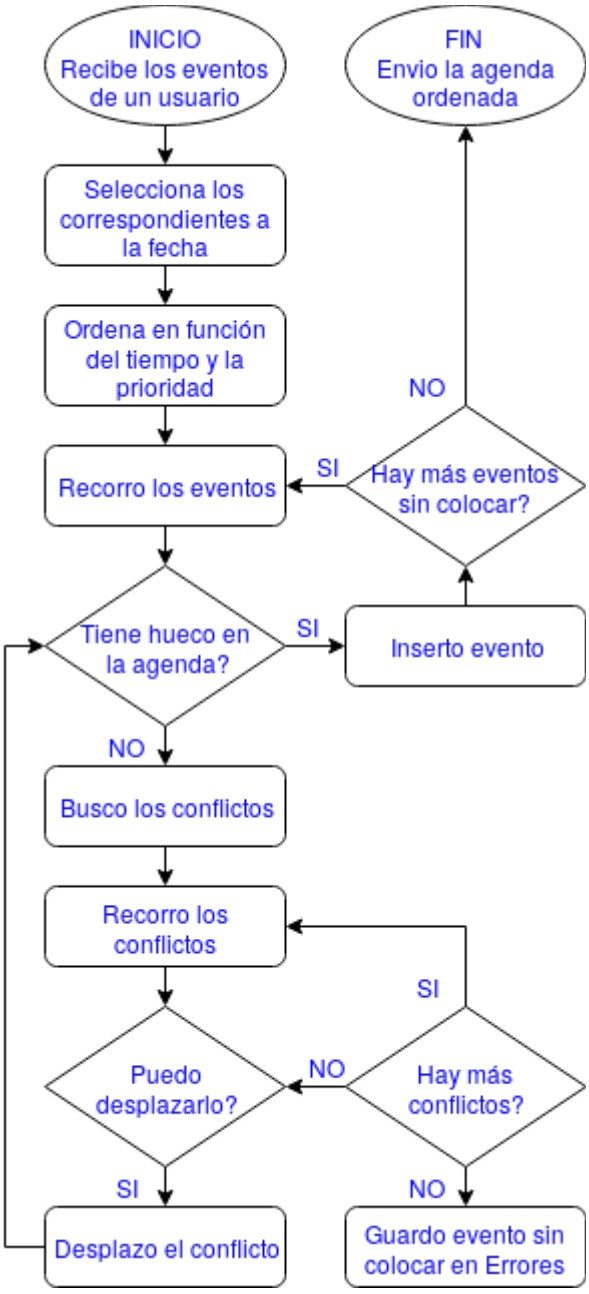
Primero pensamos en el algoritmo que ejecutábamos inconscientemente en nuestra cabeza para organizarnos el día. Y sacamos varios criterios para aplicarle a los eventos:

- **Prioridad:** Es necesario para que el algoritmo tenga una buena preferencia a la hora de conflictos y elija el evento más importante.
- **Duración:** Necesario para rellenar el hueco correspondiente en la agenda.
- **Rango horario:** Una hora de inicio y otra final para darle flexibilidad al algoritmo a la hora de ordenar los eventos. Por ejemplo, tengo que ir a la farmacia pero tengo toda la tarde para hacerlo, me es indiferente ir a las 5 que a las 7.
- **Rango de fechas:** Así conseguimos que un evento este presente en un determinado periodo de tiempo.
- **Periodicidad:** Nos ayuda a no tener que repetir el mismo evento para diferentes fechas.
- **Lo más tarde:** Esta variable indica cuando quieres que un evento sea introducido al final de su rango horario. Esto es útil para que un mismo evento valga para diferentes situaciones. Por ejemplo, si tenemos que sacar al perro

por las mañanas en un rango horario de 7am a 11am queremos sacarlo lo más tarde dependiendo del resto de tareas que tengamos, y no todos los días a las 7 de la mañana.

Una vez que el usuario ha programado sus eventos con sus distintas características y accede a una determinada fecha o un rango de fechas como una semana o un mes. Comienza a trabajar el algoritmo de la siguiente forma:

- Recibe todos los eventos de un usuario.
- Selecciona entre los eventos los correspondientes a la fecha solicitada.
- Una vez que tiene los eventos, los ordena en función de la hora y la prioridad, para ir insertando en la desde los eventos importantes hasta los secundarios.
- Recorre los eventos y comprueba si hay hueco en la agenda para insertarlo.
- Si hay hueco, rellena los huecos correspondientes en la agenda y vuelve a recorrer los eventos.
- Si no hay hueco, busca los eventos con los que entra en conflicto, los que se encuentran dentro de su rango horario.
- Intenta desplazar los eventos conflictivos e insertar de nuevo el evento, si no lo consigue lo añade a errores.
- Cuando no queden eventos sin colocar devuelve la agenda ordenada a la pagina web.



Código Fuente

Este algoritmo ha sido desarrollado en el lenguaje de programación, Python3. Cuyo conocimiento lo hemos adquirido este curso en la asignatura Lenguaje de Marcas. Sin embargo para este algoritmo hemos tenido que ampliar conocimientos y estudiar librerías más especificas como:

- **Datetime:** Necesaria para manejar los datos de tiempo, poder operar con ellos y conseguir aspectos interesantes como el día de la semana al que corresponde.
- **MySQL:** Necesaria para la comunicación con la base de datos.

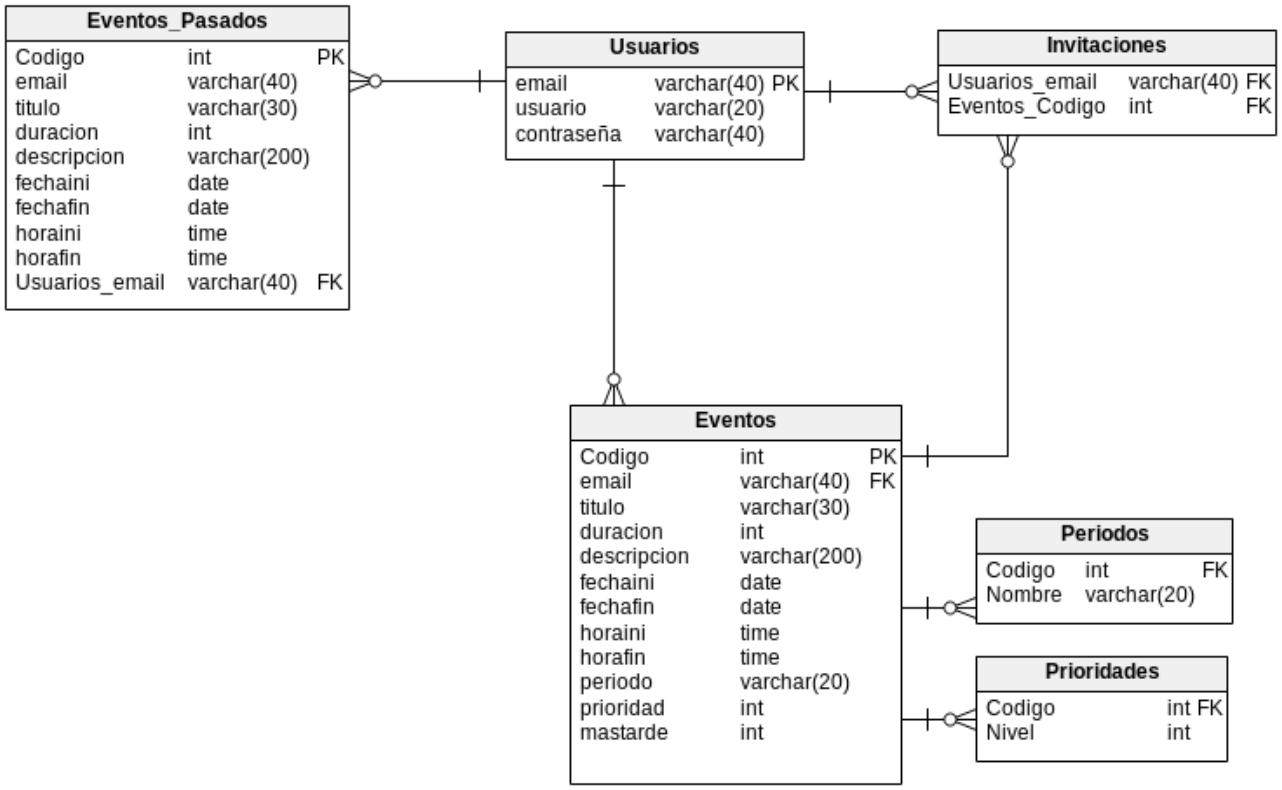
El código esta subdividido por funciones (secundarias) llamadas por la función principal. A continuación explicamos las diferentes funciones también documentadas en el repositorio de Github.

- **IngresarEventos:** Es la encargada de solicitar los eventos a la base de datos y guardarlos con una estructura especifica en formato de diccionario.
- **SelecEventos:** Selecciona entre todos los eventos los correspondientes a la fecha solicitada.
- **GenerarTablaDia:** Genera una lista del día con todos los segmentos de tiempo predefinidos, en nuestro caso cada segmento es de 5 minutos.
- **TablaTemPri:** Guarda los eventos en una tabla ordenados en función de la prioridad y el tiempo.
- **BuscarSeg:** Función auxiliar que calcula en que posición de la tabla día va un determinado evento.
- **Comprobar:** Comprueba si hay hueco para un determinado evento en la tabla día.
- **GenerarCandidatos:** Crea una lista con todos los posibles candidatos de un evento. En función de su rango horario y su duración, el evento puede tener más de una hora de inicio posibles.
- **CalcularCandidatos:** Función auxiliar de la función *GenerarCandidatos* que indica el numero de candidatos que hay que generar.
- **BusquedaProfunda:** Es la encargada de gestionar la mayoría de funciones para calcular la agenda probando diferentes opciones.
- **SelecCandidato:** Busca en la tabla tempri el candidato correspondiente comprobando que puede insertarlo en la agenda y lo retorna con un indicador a 1, en caso contrario devuelve el evento con el indicador a 0.
- **Rellenar:** Rellena los segmentos correspondientes a un evento en la tabla día.
- **Borrar:** Borra los segmentos correspondientes a un evento en la tabla día.
- **Conflictos:** Busca todos los conflictos de un evento fallido para intentar reestructurar la agenda.

3.2 Equipo de Base de Datos:

Modelo Entidad/Relación

Una vez analizado el problema que queríamos resolver y teniendo en cuenta los objetivos planteados, el equipo de Base de Datos comenzó a trabajar en el modelo de datos de la aplicación. Tras varias reuniones se decidió que el modelo Entidad/Relación que íbamos posteriormente a diseñar sería el siguiente:



- **Usuarios**: Tabla donde estarán almacenados los datos del usuario en la cuál usamos como clave primaria el **email** del usuario ya que será el método de identificación en la aplicación y no estará duplicado.
- **Eventos**: Aquí almacenamos la programación de un evento asociado al usuario que la ha creado. Nuestra aplicación consultará está tabla para calcular un resultado y mostrarnos los eventos. En esta tabla guardamos la siguiente información:
  - Un código, para identificar el evento.
  - El título y la descripción del evento.
  - La duración, expresada en minutos.
  - Las fechas entre las cuales este evento va a estar activo.
  - Las horas entre las que puedo realizar el evento.
  - El periodo de realización, puede ser fijo, diario, semanal, mensual,…
  - La prioridad, un número del 1 al 3 que servirá para repartir de forma más optima los eventos.
  - Mastarde, para indicar que un evento sea introducido al final de su rango horario.

- **Periodos**: Hemos decidido crear una tabla específicamente para la frecuencia con la que se repiten los eventos para así poder siempre controlar que en la tabla eventos solo haya un código de periodicidad que ya exista en esta tabla. De modo que a la hora de crear nuevas frecuencias solo deberíamos añadir un registro a esta tabla.
- **Prioridades**: De igual modo que con los periodos también hemos creado una tabla para las prioridades.
- **Eventos Pasados:** Esta tabla nos permite guardar los eventos ya repartidos en fechas pasadas. Con esto logramos que el reparto de eventos se haga sólo hacía el futuro. Y si queremos ver las tareas que hemos realizado en días anteriores se cogerán los datos de esta tabla.
- **Invitaciones:** Esta tabla nos va a permitir la función de sincronizar agendas. Un usuario va a poder invitar a otro usuario que realicen un evento conjunto, por lo que ese evento habrá que repartirlo para cada uno de los usuarios, y haciendo que coincidan en hora.

En la primera versión de nuestra aplicación y atendiendo a las funciones que vamos a desarrollar, sólo vamos a utilizar las tablas de usuarios y de eventos.

Diseño de la base de datos

A partir del modelo de datos, el equipo de bases de datos decidió utilizar el Gestor de Base de Datos **mariadb** para guardar la información de la aplicación. Se realizo la instalación de un servidor de base de datos para desarrollo, para que los demás equipos pudieran hacer uso de ella. Para finalizar se crearon los scrips SQL que permitieron la creación de las tablas y la inserción de registros de ejemplos para empezar a probar.

En al fase final del proyecto, nuestro equipo a desarrollado una librería de acceso a la base de datos que ha sido utilizada por el equipo de desarrollo y diseño para el acceso a la base de datos.

3.3 Equipo de Diseño Web:

Hemos decidido desarrollar nuestra aplicación web con el **microframework flask** de python. Es la tecnología que estudiamos en el módulo de "Lenguajes de Marcas" en nuestro ciclo formativo para el desarrollo de aplicaciones web.

Flask es un "micro" framework escrito en Python y concebido para facilitar el desarrollo de aplicaciones Web bajo el patrón Modelo-Vista-Controlador.

Podemos definir framework como un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. En general los framework están asociado a lenguajes de programación (Ruby on Rails (Ruby), Symphony (PHP),...).

Las ventajas tiene utilizar un ‘framework’ pueden ser:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

- Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

La elección de Flask se debe a sus características, de las cuales las más interesantes son:

- Flask es un "micro" framework: se enfoca en proporcionar lo mínimo necesario para que puedas poner a funcionar una aplicación básica en cuestión de minutos. Se necesitamos más funcionalidades podemos extenderlo con las Flask extensions.
- Incluye un servidor web de desarrollo para que puedas probar tus aplicaciones sin tener que instalar un servidor web.
- También trae un depurador y soporte integrado para pruebas unitarias.
- Es compatible con python3, por lo tanto podemos usar la codificación de caracteres unicode, y 100% compatible con el estándar WSGI.
- Buen manejo de rutas: Con el uso de un decorador python podemos hacer que nuestra aplicación con URL simples y limpias.
- Flask soporta el uso de cookies seguras y el uso de sesiones.
- Flask se apoya en el motor de plantillas Jinja2, que nos permite de forma sencilla renderizar vistas y respuestas.
- Flask no tiene ORMs, wrappers o configuraciones complejas, eso lo convierte en un candidato ideal para aplicaciones ágiles o que no necesiten manejar ninguna dependencia. Si necesitas trabajar con base de datos sólo tenemos que utilizar una extensión.
- Este framework resulta ideal para construir servicios web (como APIs REST) o aplicaciones de contenido estático.
- Flask es Open Source y está amparado bajo una licencia BSD.
- Puedes ver el código en Github, la documentación es muy completa y te puedes suscribir a su lista de correos para mantenerte al día de las actualizaciones.

Nuestra aplicación web

Nuestra aplicación web nos permite visualizar los eventos que reparte de forma inteligente nuestro algoritmo de reparto de eventos candidatos a partir de la programación de eventos.

Las distintas funciones que vas a poder realizar en nuestra aplicación web van a ser las siguientes:

- Los usuarios se va a poder registrar en nuestra aplicación y van a poder acceder a ella mediante una contraseña.
- Un usuario va a poder dar de alta nuevos eventos en su programación de eventos.
- Se puede visualizar todos los eventos programados por un usuario.
- Se pueden modificar o borrar los eventos programados.
- Se puede visualizar el calendario donde se muestra los eventos repartidos por nuestro algoritmo en distintos formatos: mensual, semanal, diario y agenda.
- Se puede navegar en el calendario, visualizar distintos meses, semanas o días.
- Se puede ver los eventos que han entrado en conflicto y no se han podido repartir.

Código fuente de la aplicación

Los ficheros que forman parte de nuestra aplicación son los siguientes:

1. **app.py:** Es el programa principal de nuestra aplicación web escrita en Flask. En este fichero declaramos la lógica (controlador) de nuestra aplicación, indicando las diferentes rutas de la aplicación y las funciones que se ejecutan cada vez que accedemos a una de ellas. Las principales rutas a las que se puede acceder a nuestra aplicación son las siguientes:

    - **/:** Ruta de entrada a la aplicación, que muestra la página html de bienvenida.
    - **/login:** Ruta que nos muestra la página de login, donde un usuario puede acceder a la aplicación por medio de su correo electrónico y su contraseña. Los usuarios y contraseñas estarán guardadas en una tabla de la base de datos. Una vez que un usuario accede se crea una variable de sesión que nos permite comprobar que un usuario está conectado.
    - **/registro:** Es la ruta que muestra la página html de registro, donde el usuario se puede dar de alta, introduciendo su nombre de usuario, su correo y la contraseña. Cuando se introducen los datos, se validan y si todo es correcto se da de alta al usuario en la base de datos.
    - **/logout:** Ruta que permite al usuario salir de su sesión.
    - **/eventos:** Al acceder a esta ruta podemos ver los eventos que tiene programado el usuario que está conectado. A esta ruta no se puede acceder si el usuario no ha iniciado su sesión. Desde esta página podemos modificar o borrar cualquier evento.
    - **/eventos/add:** Ruta que nos permite añadir un evento a nuestra programación.
    - **/eventos/del/<cod. evento>:** Nos permite borrar el evento indicado de nuestra programación.
    - **/eventos/edit/<cod. evento>:** Nos permite modificar el evento indicado de nuestra programación.
    - **/calendar:** Esta ruta nos permite visualizar el calendario con los eventos del usuario que está en la sesión, que nuestro algoritmo ha repartido. En esta ruta se muestra una página html que añade un script javascipt que permite visualizar un calendario (fullcalendar.io). A este componente es necesario suministrarle los eventos que tiene que mostrar, para ello, lo hemos configurado para que haga una petición a la ruta /data. En la petición envía la fecha inicial y la final de los días que va a mostrar, y obtiene los datos de los eventos a mostrar representados con el formato json.
    - **/data:** Es la ruta de nuestra aplicación que utiliza el componente javascript fullcalendar para obtener los eventos a mostrar. Esta ruta devuelve la información de los eventos en formato json.

2. **agenda.py:** Librería de funciones python que realizan el reparto de eventos candidatos a partir de la programación de eventos.

3. **fullcalendar.py:** Librería python que contiene la función que obtiene la información necesaria para que el script fullcalendar sea capaz de visualizar los eventos. Recibe la fecha inicial y final de los días en las que tiene que generar los eventos y utiliza las funciones de la librería `agenda.py` para seleccionar los eventos diarios que debe mostrar.

4. **basedatos.py:** Librería python que tiene la función que nos permite realizar las consultas a la bse de datos MySQL.

Diseño de la página web

El framework Flask utiliza plantillas html jinja2 para generar las páginas web dinámicas que se van a visualizar. El el directorio **templates** encontramos las distintas plantillas que necesita nuestra aplicación:

- **base.html:** Plantilla base de la aplicación, todas las demás plantillas van a ser una herencia de esta. Posee la parte común de todas las páginas, como la sección `head` del documento HTML y el píe de página.
- **menu.html:** La sección de navegación de nuestra página se encuentra en esta plantilla que se incluye en la anterior.
- **index.html:** Página principal de la aplicación, se extiende a partir de *base.html* y muestra información de la aplicación, con las distintas opciones que tenemos a nuestra disposición.
- **login.html:** Página web que muestre el formulario para que un usuario pueda autentificarse.
- **registro.html:** Página web que muestre el formulario para que un usuario pueda registrarse en la aplicación.
- **eventos.html:** Página web que visualiza los eventos programados para un determinado usuario.
- **addeventos.html:** Página web que visualizar el formulario para dar de alta o modificar une evento en la programación de un usuario.

Otros ficheros

En la carpeta **static** encontramos el contenido estático de nuestra aplicación. Podemos encontrar las siguientes carpetas:

- **css:** Hemos elegido como framework para desarrollar la hoja de estilo bootstrap v4. Aunque los ficheros de hoja de estilo y javascript que necesita bootstrap se añaden desde un sitio de internet en la plantilla **base.html.**

Hemos incluido algunas hojas de estilos adicionales para el diseño de nuestras páginas web.

- **fullcalendar:** Todos los script de *fullcalendar.io* que hemos descargado de *https://fullcalendar.io/download.*
- **img:** Directorio donde se encuentran las imágenes que visualiza nuestra aplicación.

Por último indicar que en el fichero **requirement.txt** se encuentran las librerías python necesarias para que la aplicación funcione.

3.4 Despliegue de nuestra aplicación web en un entorno de producción

Vamos a desplegar nuestra aplicación web desarrollada con flask en un servidor LAMP (Linux+Apache2+mariaDB+python) en un sistema operativo GNU/Linux Debian 9.

Instalación de los servicios

En el servidor hemos instalado los siguientes servicios:

- **apache2:** Es el servidor web más utilizado en la actualidad, hemos instalado los módulos necesarios para que sea capaz de servir aplicaciones escitas en python.
- **MariaDB**: Base de datos que hemos utilizado en el entorno de desarrollo, y que vamos a usar también en el entorno de producción para guardar la información de nuestra aplicación.

Configuración de la infraestructura

Hemos creado un entorno virtual python que nos permite hacer una instalación aislada de las librerías necesarias para el funcionamiento de la aplicación y que podemos encontrar en el fichero *requirements.txt* de nuestra aplicación. A partir de este entorno hemos configurado el servidor web para que utilice el protocolo WSGI (Web Server Gateway Interface), que es una especificación de una interface simple y universal entre los servidores web y las aplicaciones web o frameworks desarrolladas con python.

Por último hemos creado en el servdor DNS del instituto un nuevo nombre con el que accederemos a nuestra aplicación en producción:

http://protato.gonzalonazareno.org

4.- Colaboración

El presente proyecto ha sido planificado y elaborado por un grupo de alumnos del primer curso del Ciclo Formativo de "Administración de Sistemas Informáticos". Durante las distintas etapas del desarrollado del proyecto, y teniendo en cuenta que iba a hacer un trabajo colaborativo, se ha tenido en cuenta los siguientes aspectos:

El grupo al completo ha realizado la etapa de análisis de requisitos:

Decidir que aplicación íbamos a desarrollar, y posteriormente hacer el análisis de los requisitos y la toma de decisiones de diseño. Esta etapa ha sido la que más tiempo nos ha llevado, ya que hemos necesitado varias jornadas de trabajo para ponernos de acuerdo en el problema que queríamos resolver.

Creación de varios equipos de trabajo:

- **Equipo de diseño de la base de datos**: Encargado de definir el modelo de datos que iba a necesitar para guardar la información con la que vamos a trabajar. Posteriormente el encargado de implantar el gestor de base de datos para guardar la información.

- **Equipo de diseño de la página web**: Responsable de diseñar el interfaz de usuario, y las distintas opciones que va a tener nuestra aplicación.

- **Equipo de desarrollo de la aplicación python**: Encargado de realizar el programa que permite trabajar con los datos de la aplicación y que posteriormente se va a visualizar en la aplicación web.

En la creación de los equipos de trabajo se tuvo en cuenta lo siguiente:

- Cada equipo tendrá una personal responsable *(facilitador)*.

- Los equipos son abiertos, todos los miembros deben estar informados de los que demás hacen, para cuando sea necesario puedan cambiar de equipo.

- Durante el desarrollo del proyecto se pueden crear de nuevos equipos de trabajo según las necesidades.

Utilización de metodologías ágiles, en concreto scrum para el desarrollo del proyecto:

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Hemos realizado varias iteraciones (sprint), donde en un principio se han asignado tareas a los distintos equipos de trabajo, que se han evaluado al finalizar la misma. Las conclusiones de cada iteración nos han permitido definir los nuevos objetivos que debemos alcanzar y el reparto de tares para la siguiente iteración.

Las herramientas que vamos a utilizar para el desarrollo del proyecto serán las siguientes:

- **Github**: Como repositorio de la documentación y el código.
- **Redmine**: Como gestor de proyecto.
- **Trello**: Para implementar el tablero Scrum.
- **Slack**: Como herramienta de comunicación.

5.- Resultados

Para todo el equipo, del instituto Gonzalo Nazareno, los resultados han sido bastante satisfactorios. Hemos conseguido una primera versión funcional de la aplicación.

Hemos conseguido una aplicación capaz de organizar una agenda de forma simple y eficiente. Accesible mediante una pagina web, una plataforma multiusuario donde pueden introducir las diferentes tareas del día a día como: rutinas diarias, rutinas semanales, eventos fijos...

Las cuales son organizadas de forma eficiente y mostradas en una agenda dinámica gestionada por un framework ajeno (fullcalendar) que hemos incorporado en nuestra pagina.

Para las próxima versiones nos queda pendiente habilitar la función para sincronizar agendas entre usuarios, la función de eventos pasados para dejar constancia de un histórico con los eventos realizados y la creación de una API Restful que respondería a las consultas de una aplicación móvil.

6.- Conclusiones

El equipo del **I.E.S Gonzalo Nazareno** estamos muy agradecidos a la empresa **ec2ce** por organizar e incluirnos en este fantástico concurso el cual nos ha emocionado al vernos envueltos en un proyecto enfocado de forma profesional. El mero hecho de haber podido tener la oportunidad de emprender nuestro propio proyecto ha sido muy satisfactorio para todos nosotros, pues es algo que ha requerido de nuestra propia imaginación y que nos permite salir de la rutina del ámbito docente. Esto nos ha proporcionado la oportunidad de buscar ideas propias y de poder llevarlo a cabo en un proceso tan real y profesional.

En cuanto al trabajo en equipo, ha habido una buena integración entre los distintos componentes de éste. Hemos integrado nuestros conocimientos de forma conjunta y eso ha hecho que incrementamos nuestros conocimientos individuales.

Por otro lado, nos hemos adentrados en conocimientos más específicos que estaban fuera de nuestra programación docente, aprendiendo aún más de lo esperado.

En definitiva, estamos muy contentos con los resultados obtenidos y creemos que hemos desarrollado una aplicación muy interesante y novedosa que podría tener muchas aplicaciones en el futuro, estamos seguros de que esta aplicación sería muy útil tanto a nivel doméstico como empresarial.