Instituto Tecnológico de Costa Rica
Proyecto de Ingeniería de Software
2016106261 - José Carlos Montoya Pichardo

Bitácora Semana #4

Miércoles 26 de Diciembre de 2018

9:00 am: En GitHub se crea la rama ‘adm’ para gestionar la aplicación web de administración.
9:10 am: Se configura Bootstrap para brindar comportamiento Responsive a la app. 9:15 am : Se configura Angular App Routing y se crean los componentes de la página (no se configuran).
9:45 am: Se soluciona un problema de la rama al subir el proyecto al Git. Fue necesario borrar la rama y crearla nuevamente.
10:20 am: Diseño gráfico de la página para crear Noticias.
3:00 pm: Conexión a la base de datos para el almacenamiento de noticias. Se configura el método CREATE.
3:30 pm: Se configura el método GET. Solamente se solicitan las noticias que se hayan publicado durante la última semana.
4:00 pm: Se configura el método UPDATE.
4:30 pm: Se configura el método DELETE.
5:00 pm: Se configuran snackbars para mostrar mensajes de resultados para operaciones CRUD.
5:15 pm: Se configura un diálogo genérico para confirmación de acciones de usuario.
11:55 pm: Se configura un Router Resolver para que la página no se muestre durante el tiempo de carga de datos desde BD. También se configura una animación que indica que se encuentra en el proceso de carga de datos. También se configura una carpeta para gestionar componentes que son compartidos por varios elementos de la página, por ejemplo, diálogos.

Jueves 27 de Diciembre de 2018

9:00 am - 10:55 am: Modificación de la barra de navegación para otorgar comportamiento Responsive.
11:10 am - 12:10 pm: Solución a error de overflow de etiquetas en la barra de navegación, corrección del divider horizontal en tarjeta de administración de Noticias y arreglo de márgenes en cuerpo de página.
2:00 pm - 3:00 pm: Diseño gráfico de la página para crear Eventos.
3:00 pm- 5:30 pm: Diálogo para la solicitud de fecha y hora. Se realizó una configuración especial para tomar la fecha en formato UTC, ya que Angular, por defecto,toma la hora con diferencia según la zona horaria.
5:30 pm - 6:30 pm: Configuración del formulario de eventos. Los controles de solicitud de fecha están en una lista. Se hace el primer POST de evento con éxito.
7:55 pm - 8:07 pm: Se configura el método UPDATE.
8:07 pm - 8:12 pm: Se configura el método DELETE.
8:15pm - 8:30pm: Se configura el método GET. El ordenamiento de eventos se realiza de acuerdo a la fecha de inicio. Los eventos cuya fecha de finalización es menor a la actual no

Instituto Tecnológico de Costa Rica
Proyecto de Ingeniería de Software
2016106261 - José Carlos Montoya Pichardo
se solicitan a la BD.

Domingo 30 de Diciembre de 2018

9:30 pm - 11:40 pm : Se diseña gráficamente el componente Login.

Miércoles 2 de Enero de 2019

10:00 am - 11:00 am: Se configura un backend Python para la escucha de mensajes enviados desde Arduino. Las banderas de los contadores se encuentran en un objeto Contador que recibirá peticiones desde este backend, por cada petición se realizará una modificación a estas banderas y se hará un respaldo en un archivo JSON.
11:00 am - 2:00 pm: Se configura un backend Python para escuchar peticiones desde las páginas web (pública y de administración) utilizando Flask. Además se configura un proceso programado para que realice el envío, a la BD, del número de vehículos que ingresaron durante el día.
3:00 pm - 6:00 pm: Se intentó configurar el backend de las páginas web para que el método GET se comporte como un SSE y que el envío de las banderas del objeto Counter sea automático y así las páginas se actualicen en tiempo real. Sin embargo hubo problemas para completar esta funcionalidad. 8:00 pm - 10:00 pm: Se terminaron de configurar otros métodos para la modificación de banderas de Counter con peticiones enviadas desde las páginas Web.

Jueves 3 de Enero de 2019

8:00 am - 10:00 am: Se decidió utilizar Flask SSE para obtener la funcionalidad de envío de datos mediante un canal Stream. Este componente requiere la utilización de Redis y Gunicorn (para permitir múltiples hilos). Debido a la naturaleza de Gunicorn la configuración manejada hasta el momento no iba a funcionar.
10:00 am - 12:00 pm: Reunión con el cliente para la recopilación de requerimientos.
12:00 pm - 3:00 pm: Se decidió desechar el objeto Counter y poner sus banderas en un script aparte y separar los dos backend como scripts individuales (antes eran objetos de un mismo proceso) . Para coordinar la comunicación de ambos scripts se decidió utilizar ZeroMQ por lo que fue necesario realizar la instalación, configuración y leer documentación para conocer su funcionamiento.
3:00 pm - 7:00 pm: Después de realizar las configuraciones anteriores el sistema seguía presentando problemas. Se descartó el uso de Flask SSE y se intentó utilizar solamente Redis para gestionar el SSE. Se logró implementar el Stream pero no del todo bien.

Viernes 4 de Enero de 2019

6:00 pm - 9:00 pm: Se inicia la redacción del documento de requerimientos.
9:00 pm - 11:59 pm: Se decide utilizar Flask SSE nuevamente. Fue necesario leer documentación de Gunicorn para entender el funcionamiento.

Instituto Tecnológico de Costa Rica
Proyecto de Ingeniería de Software
2016106261 - José Carlos Montoya Pichardo

Sábado 5 de Enero de 2019

12:00 am - 1:30 am: Se implementó el uso de Flask SSE + Gunicorn + Gevent + Redis para lograr que los SSE funcionen. Se decidió que las banderas de los contadores se ahora se manejan en el backend web, por lo que el uso de ZeroMQ fue descartado.
1:30 am - 2:00 am: Se agregó más contenido al documento de requerimientos.
8:00 am - 12:00 pm: Se configuró el inicio de sesión de usuario utilizando la tecnología de Loopback (modelo User preconfigurado) y se realiza la gestión del Access Token utilizando JWT desde Angular.
1:00 pm - 2:00 pm: El cliente solicitó la inclusión de TinyMCE para editar noticias y eventos y la funcionalidad de programar la publicación de las noticias y eventos. Solamente se implementó la segunda solicitud.
2:00 pm - 5:00: Se configuró la pantalla de gestión de usuarios. Se logró implementar la funcionalidad para crear usuarios pero las funcionalidades para modificar y borrar usuarios no se pudieron cumplir debido a limitaciones de seguridad del modelo User de Loopback. Se trató de solucionar esto de varias maneras pero ninguna tuvo éxito.
5:00 pm - 8:30 pm: Redacción del documento de arquitectura de software.
10:00 pm - 11:00 pm: Aplicación de formato final a los documentos para entregar.