Actividad 5. Aplicación Privada (continuación)

Fecha de entrega: 15/11 a las 23:59

Puntaje máximo esperado: 40 pts

Puntaje entrega tarde: 30 pts

2.1 Módulo de centro de ayuda social

En este módulo se realiza la administración de los centros de ayuda cargados en el sistema, ya sea desde la aplicación privada o pública. Si un centro es cargado desde la aplicación pública quedará pendiente de aprobación[1], es decir, un usuario con el rol de **Operador** o **Administrador**, deberá aprobar la carga para que el centro pueda visualizarse en mapas, listados y aplicación pública.

2.1.1 CRUD de centro de ayuda social

El módulo debe brindar **al menos** la siguiente funcionalidad:

- CRUD de centro de ayuda social.
- Se deben listar los campos que permitan ver la información relevante (consultar con el ayudante en caso de alguna duda).
- Se deben poder realizar búsquedas, **al menos** por los siguientes campos:
    - nombre del centro de ayuda social (texto).
    - estado: pendiente, aceptado o rechazado (select).

  Los resultados también deben estar paginados tomando los parámetros de cantidad de páginas del **módulo de configuración**.
- Se debe poder realizar la acción de aceptar o rechazar un centro de ayuda social nuevo agregado. Un usuario con rol **Operador** es quien generalmente realiza el proceso de certificación validando si los datos del centro de ayuda social agregado son reales. Además el administrador podrá realizar esta acción en caso que se requiera.

A continuación se definen los roles necesarios para el resto de las acciones:

- index, show, update, create: **Operador, Administrador**
- destroy: **Administrador**

Los datos necesarios para cada centro son:

- Nombre*: nombre del centro de ayuda social (text)
- Dirección*: dirección del centro de ayuda social (text). Este dato será utilizado para visualizarlo en un mapa.
- Teléfono*: número de teléfono (text)
- Hora de apertura*: hora de apertura (time)

[1] La funcionalidad para la carga de centros de ayuda se trabajará en la Actividad 6

- Hora de cierre*: hora de cierre (time)
- Tipo de centro de ayuda social.*: listado de tipos de centros[2] (select)
- Municipio.*: listado de municipios[3] (select)
- Web: sitio web del centro de ayuda social (text)
- Email: dirección de correo electrónico (text)
- Estado: publicado o despublicado
- Protocolo de vista: archivo formato PDF
- Coordenadas: coordenadas geográficas del centro de ayuda[4]

Para la carga de coordenadas en el formulario deberán hacer uso de un componente de mapa que permita elegir el punto donde está ubicado el Centro de Ayuda y de esta forma obtener las mismas. Recomendamos hacer uso de la siguiente librería.

2.1.2 API de Centro de ayuda social

Se deberán desarrollar los servicios de la **API** que serán accedidos desde de la aplicación pública[5] con el cliente Vue.js.
La API de centros de ayuda social tendrá los siguientes *endpoints*[6]:

Path	/centros
Method	GET
Description	Listado de centros de ayuda social. Este servicio permite obtener el listado completo de los centros de ayuda social aprobados para publicación. Este listado también debe estar paginado teniendo en cuenta el número que se carga en la configuración para la cantidad de registros por página.
Success response	**Status code**: 200 **Body**:

[2] Listado de centro de ayuda social publicados que se encuentran en nuestra base de datos
[3] Listado de municipios que obtenemos de la API de Referencias
[4] Será necesario tener las coordenadas del eje latitud y longitud para poder ubicar los puntos en los distintos mapas
[5] A desarrollar en la Actividad 6
[6] Un *endpoint* es un punto de acceso a un servicio para una Arquitectura Orientada a Servicios

	{
	"centros":
	[
	{
	"nombre": "Iglesia Sagrado Corazón de Jesús",
	"direccion": "Diagonal 73 nro 1032",
	"telefono": "221 - 5139019",
	"hora_apertura": "09:30",
	"hora_cierre": "18:00",
	"tipo": "Institución religiosa",
	"web": "",
	"email": ""
	},
	{
	"nombre": "Merendero Todos por una Sonrisa",
	"direccion": "Calle 88 nro 1912, Altos de San Lorenzo",
	"telefono": "221 - 5930941",
	"hora_apertura": "15:00",
	"hora_cierre": "18:00",
	"tipo": "Merendero",
	"web": "",
	"email": ""
	}
	],
	"total": 2,
	"pagina": 1
	}

Error response	**Status Code:** 500 Internal Server Error

Path	/centros/:centro_id
Method	GET
Description	Obtener un centro de ayuda social. Este servicio permite obtener el centro de ayuda social aprobado para publicación, que corresponde al identificador pasado por parámetro.

Success response	**Status code**: 200 OK
	Body:
	{
	"atributos":
	{
	"nombre": "Merendero Todos por una Sonrisa",
	"direccion": "Calle 88 nro 1912, Altos de San Lorenzo",
	"telefono": "221 - 5930941",
	"hora_apertura": "15:00",
	"hora_cierre": "18:00",
	"tipo": "Merendero",
	"web": "",
	"email": ""
	}
	}

Error response	**Status Code:** 500 Internal Server Error
Error response	**Status Code:** 404 Not Found

Path	/centros
Method	POST
Description	Cargar un centro de ayuda social. Este servicio permite cargar un centro de ayuda social por medio de la API. Se deberán enviar los mismos campos que se definieron anteriormente para la carga.
Request Body	**Body**: `{ "nombre": "Merendero Todos por una Sonrisa", "direccion": "Calle 88 nro 1912, Altos de San Lorenzo", "telefono": "221 - 5930941", "hora_apertura": "15:00", "hora_cierre": "18:00", "tipo": "Merendero", "web": "", "email": "" }`
Success response	**Status code**: 201 Created **Body**: `{ "atributos": { "nombre": "Merendero Todos por una Sonrisa", "direccion": "Calle 88 nro 1912, Altos de San Lorenzo", "telefono": "221 - 5930941", "hora_apertura": "15:00", "hora_cierre": "18:00", "tipo": "Merendero", "web": "", "email": "" } }`
Error response	**Status Code:** 500 Internal Server Error
Error response	**Status Code:** 400 Bad request

2.2 Módulo turnos de asistencia a centros de ayuda

Se deberá desarrollar un módulo para la gestión de turnos que permita organizar el acceso a los centros de ayuda, intentando evitar la concurrencia masiva de personas. De esta manera se podrán respetar los protocolos de distanciamiento y limpieza que cada centro debe cumplir.

2.2.1 CRUD de turnos

En este módulo se realiza la administración de los turnos necesarios para organizar las visitas a los centros de ayuda.
Cada módulo de turnos deberá ser accedido partiendo de la vista de un centro en particular.
Los usuarios **operadores** podrán realizar estas acciones **sobre los centros asociados.**
El módulo debe brindar las siguiente funcionalidades:

- CRUD de turnos para un centro en particular.
- Inicialmente se listan los turnos de **hoy y los siguientes 2 días** ordenados por fecha y hora ascendente.
- Para la carga de un turno se requieren los siguientes datos:
    - Email de la persona que solicita el turno.
    - Bloque del turno (tiene que ser uno disponible).
    - Día del turno.
- Se deben poder realizar búsquedas/filtros, **al menos** por los siguientes campos:
    - nombre de centro (texto).
    - email de persona que pidió el turno (select).

  Los resultados también deben estar paginados tomando los parámetros de cantidad de páginas del **módulo de configuración**.

Para simplificar la lógica de los turnos vamos a describir la disposición de turnos. La misma será fija e igual para todos los centros:

- Los turnos se darán en bloques de 30 minutos.
- Los turnos se brindarán en la franja horaria de 9hs a 16hs.
- No se distinguen los fines de semana o feriados. Todos los días son lo mismo.

2.2.2 API de turnos

Se deberán desarrollar algunos servicios públicos que permitirán realizar la solicitud de los turnos:

Path	/centros/:id/turnos_disponibles/?fecha=<fecha>
Method	GET
Description	Este servicio permite obtener el listado de los turnos disponibles para un centro de ayuda en un día en particular. En caso de no especificar una fecha, el sistema deberá devolver la disponibilidad de turnos para el día consultado. No es necesario paginar los resultados.
Success response	**Status code**: 200 **Body**: { "turnos": [{ "centro_id": 1, "hora_inicio": "09:30", "hora_fin": "10:00",

	"fecha": "2020-10-10"
	},
	{
	"centro_id": 1,
	"hora_inicio": "10:00",
	"hora_fin": "10:30",
	"fecha": "2020-10-10"
	}
	]
	}

Error response	**Status Code:** 500 Internal Server Error

Path	/centros/:id/reserva
Method	POST
Description	Realiza la reserva de un turno para un centro de ayuda en particular. El servicio debe verificar que no se agreguen dos turnos para el mismo centro y con el mismo bloque de horario en un mismo día.

Request Body	**Body**:
	{
	"centro_id": 1,
	“email_donante": "juan.perez@gmail.com",
	"telefono_donante": "221 - 5930941",
	"hora_inicio": "15:00"
	"hora_fin": "18:00",
	"fecha": "2020-10-10"
	}

Success response	**Status code**: 201 Created
	Body:
	{
	"atributos":
	{
	"centro_id": 1,
	"email_donante": "juan.perez@gmail.com",
	"telefono_donante": "221 - 5930941",
	"hora_inicio": "15:00"
	"hora_fin": "18:00",
	"fecha": "2020-10-10"
	}
	}

Error response	**Status Code:** 500 Internal Server Error
Error response	**Status Code:** 400 Bad request

3.1 Acceso a API de Referencias

En la Provincia de Buenos Aires se exceptúa del cumplimiento del "aislamiento social, preventivo y obligatorio" y de la prohibición de circular, a las personas afectadas a las actividades, servicios e industrias indicados en el ANEXO I (IF-2020-46207785-APN-SCA#JGM), actividades exceptuadas.
Dependiendo del nivel de contagio de la población, cada Municipio se encuentra en una fase determinada (1, 2, 3, 4 o 5).
Las fases y las actividades exceptuadas podrán obtenerse vía servicio consultando la API de Referencias de la Cátedra. A su vez podrán consultarse vía servicio los datos necesarios al momento de cargar un Municipio o la fase y las actividades exceptuadas en las que se encuentra un Municipio.
Datos obtenidos el día 18/10/2020 del siguiente portal.

Datos de referencia que proveerá la API:

- Fase
- Actividad exceptuada
- Municipio

API de Referencias

URL base: https://api-referencias.proyecto2020.linti.unlp.edu.ar

Endpoints

- `/municipios`
- `/actividades`
- `/fases`
- `/municipios/:id/actividades`

Particularidades

- Todos los servicios tienen una paginación de 25 elementos por defecto
- El parámetro **page** indica qué página se solicita y **per_page** pisa la cantidad de elementos que se traen por defecto. Ejemplo:
    - https://api-referencias.proyecto2020.linti.unlp.edu.ar/municipios/4/actividades?page=2&per_page=20

Consideraciones generales

- El prototipo debe ser desarrollado utilizando Python, JavaScript, HTML5, CSS3 y MySQL, y **respetando el modelo en capas MVC**.
- El código deberá escribirse siguiendo las guías de estilo de Python
- El código Python deberá ser documentado utilizando docstrings.
- **El uso de jinja como motor de plantillas es obligatorio para la aplicación privada**.
- Se debe utilizar Flask como framework de desarrollo web para la aplicación privada.
- Se deberán realizar **validaciones de los datos de entrada** tanto del lado del cliente como del lado del servidor. Para *las validaciones del lado del servidor se deben realizar en un módulo aparte* que reciba los datos de entrada y devuelva el resultado de las validaciones. En caso de fallar el controlador debe retornar la respuesta indicando el error de validación.
- Podrán utilizar librerías que facilitan algunas de las tareas que deben realizar en el trabajo como pueden ser: conexión a servicios externos, librerías de parseo, librerías con patrones para buenas prácticas, validaciones de datos, etc. **Pero todos los miembros del equipo deben demostrar en la defensa pleno conocimiento del funcionamiento de estas librerías y una idea de cómo solucionan el problema**.
- Para la interacción con la base de datos pueden utilizar un ORM que nos permita además tener una capa de abstracción con la BD.
- No pueden utilizar un framework/generador de código para el desarrollo de la API de Centros de Ayuda o Turnos.
- Debe tener en cuenta los conceptos de Semántica Web proporcionada por HTML5 siempre y cuando sea posible con una correcta utilización de las etiquetas del lenguaje.centros
- El trabajo será evaluado desde el servidor de la cátedra que cada grupo deberá gestionar mediante Git. **NO se aceptarán entregas que no estén realizadas en tiempo y forma en el servidor provisto por la cátedra**.
- Deberán visualizarse los aportes de cada uno/a de los/as integrantes del grupo de trabajo.
- El/la ayudante a cargo evaluará el progreso y la participación de cada integrante mediante las consultas online y el seguimiento mediante GitLab.
- Toda vista (**HTML5** y **CSS3**) debe validar contra las especificaciones de la W3C (http://validator.w3.org/ y https://jigsaw.w3.org/css-validator/ respectivamente). En esta oportunidad puede utilizar **Bootstrap** u otro framework similar. En caso de que alguna de las vistas no valide, deberá realizar un breve informe indicando cuales son los errores encontrados.
- Todas las vistas deben cumplir ser web responsive y visualizarse de forma correcta en distintos dispositivos. Al menos deben contemplar 3 resoluciones distintas:
    - < 360
    - 360 < res < 768
    - > 768
- La entrega es obligatoria. Todos y todas los/as integrantes deben presentarse a la defensa.
- El sistema no debe ser susceptible a SQL Injection, XSS ni CSRF.
- **Importante**:
    - El proyecto podrá ser realizado de modo individual o en grupos de tres o

cuatro integrantes (será responsabilidad de los y las estudiantes la conformación de los equipos de trabajo). Todos y todas los/as estudiantes cumplirán con la totalidad de la consigna, sin excepciones.

Información del servidor

Tener en cuenta que el servidor de la cátedra utiliza los siguientes servicios y versiones:

- Servidor de Base de Datos: MariaDB 10.1.44
- Intérprete Python: Python 3.6.9
- Servidor web: Apache 2.4.29 (Ubuntu)