Proyecto Semestral

## 1-3 personas

# Introducción

Construya un sistema de información para un parqueadero. El parqueadero está compuesto por varios pisos, en cada uno de los cuales se guarda un tipo de vehículo (automóviles, buses, motos). Puede haber más de un piso para un mismo tipo de vehículo. Dicho ordenamiento lo revisa periódicamente el administrador según el porcentaje de ocupación que haya tenido cada piso en el mes anterior y se puede cambiar. Todos los pisos son de igual tamaño (rectangulares), y el número de vehículos que puede parquearse en cada uno de ellos depende del tipo de vehículo.

# Requisitos

El sistema debe proveer las siguientes funcionalidades:

1. Administración de espacios de parqueadero:
   1. CRUD de pisos 3
   2. Especificar tipo de vehículo para cada piso, calcular espacios disponibles (dependiendo del tipo de vehículo se parquea en cada piso) 1
   3. Especificar tarifas por minuto por tipo de vehículo.1
   4. CRUD tipoVehiculo 2
2. Gestión de parqueos
   1. Admitir un vehículo e indicarle el piso en el cual debe parquear, dependiendo de disponibilidad de espacios. Si el parqueadero está lleno no debe permitirse acceso 1
   2. Registrar el piso utilizado por el vehículo y calcular espacios disponibles 2
   3. Registrar la salida del vehículo y calcular el cobro. 2
3. Tarifas y Espacios
   1. Mostrar las tarifas por tipo de vehículo 1
   2. Mostrar los espacios disponibles por tipo de vehículo 1
4. Seguridad y gestión de usuarios
   1. Existen tres tipos de usuario: administrador, portero y conductor. Los usuarios administrador y portero requieren autenticarse ante el sistema para utilizarlo. El conductor, no requiere autenticación
      * El portero solo puede acceder a la funcionalidad de Gestión de parqueos
      * El conductor solo puede acceder a Tarifas y Espacios
      * El administrador puede realizar todas las tareas: Administración de espacios, gestión de parqueos y Tarifas y Espacios.

# Entregas

Cada entrega debe incluir lo siguiente:

1. Documentación de diseño (lo que se debe entregar varía de una entrega a otra)
2. Código fuente
   1. Debe subirse en un archivo comprimido
   2. El código en Java Spring debe estar “limpio”, es decir, no incluir ningún archivo binario `.class`. Para ello asegúrese de ejecutar `./mvnw clean` antes de comprimir el código en un archivo
   3. De manera similar, el código en Angular no debe contener la carpeta `node\_modules`. Asegúrese de borrar dicha carpeta antes de comprimir el código en un archivo
3. Video explicativo
   1. Grabe un video explicando, paso a paso, sus respuestas. La explicación debe ser verbal, apoyada por una o varias de estas alternativas: imágenes, animaciones, grabaciones de la pantalla del computador, o grabaciones con su cámara sobre papel o tablero. Si usa esta última alternativa, asegúrese de fijar adecuadamente la cámara o celular para que no se mueva. 6
      * Si la cámara oscila o se mueve más de lo necesario, se bajará puntaje!
   2. Asegúrese que todo lo mostrado en el video tenga una resolución adecuada y que su voz se escuche claramente.
   3. El video debe comenzar con una breve explicación de los problemas a resolver y continuar con las soluciones.
   4. Suba el video a Youtube de manera no listada (Unlisted) <https://support.google.com/youtube/answer/157177?co=GENIE.Platform%3DDesktop&hl=en>
      * Alternativamente, puede usar otras plataformas: Vimeo o Microsoft Steam. En cualquier caso, asegúrese que el video no sea público y que el profesor tenga acceso.
      * Si el video es muy extenso puede dividirlo en partes, subirlos separadamente y crear un playlist
   5. Está prohibido compartir este video con cualquier otra persona que no sea su profesor

## 1a entrega

* Código fuente que implemente funcionalidad 1. Administración de espacios de parqueadero
* Documentación de diseño
  + Diagrama de navegación de la aplicación completa
  + Mockups de las pantallas de toda la aplicación. Dichos mockups deben corresponder con el diagrama de navegación
  + Documentación de la base de datos para la aplicación completa
    - Pueden usarse [modelos lógicos de datos](https://www.lucidchart.com/pages/templates/logical-data-model-example), diagramas de clases, etc.
* URL del video o playlist que explique la entrega

## 2a entrega

* Código fuente que implemente funcionalidad 2. Gestión de parqueos
* URL del video o playlist que explique la entrega

## 3a entrega

* Código fuente que implemente funcionalidades 3 y 4
* Código fuente de las pruebas de integración (sobre los servicios REST de la aplicación) y pruebas de sistema sobre la aplicación completa
* URL del video o playlist que explique la entrega

# Evaluación

Para que cada evaluación del proyecto tenga una nota > 0.0 debe cumplir con lo siguiente:

* Deben subirse a Brightspace todos los entregables requeridos
* Los entregables deben cumplir los requisitos indicados en la sección Entregables
* El código fuente debe compilar sin errores
* Una vez compilados, los programas y las pruebas (si aplican) se deben ejecutar sin errores

Si lo anterior no se cumple habrá una penalización en la nota.

Cada entrega se evaluará según las siguientes tablas.

1a entrega

|  |  |  |
| --- | --- | --- |
| Persistencia | Plantillas / interfaz | Componentes Spring |
| 60.00% | 20.00% | 20.00% |

2a entrega

|  |  |  |  |
| --- | --- | --- | --- |
| Components | Routes | Comunicación REST | Backend |
| 30.00% | 30.00% | 20.00% | 20.00% |

3a entrega

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de datos | Seguridad | Pruebas backend | Pruebas frontend |
| 25.00% | 25.00% | 25.00% | 25.00% |