Lea la prueba y teniendo en cuenta su disponibilidad de tiempo, informe el día y hora en la que máximo enviará las respuestas. Note que en el transcurso del tiempo podrá enviar preguntas que le serán respondidas con la mayor prontitud. Si por cualquier motivo no puede terminar toda la prueba en el tiempo que escogió, envíe la parte que tenga y el motivo por el cual no pudo completarla. Una adición al tiempo de entrega podría ser otorgada.

SPA CHALLENGE

1. Marvel Challenge (70 puntos): La primera parte de este SPA CHALLENGE consiste en desarrollar un SPA que permita encontrar Personajes de Marvel desde un buscador, también debe permitir ver el detalle de los cómics asociados a los personajes.

Para desarrollar te enviaremos una carpeta comprimida donde encontrarás todos los recursos que vas a utilizar para esta prueba. Para los diseños que no se encuentran especificados te lo dejamos a tu imaginación.
El buscador de Personajes debe consumir el MARVEL API REST, en este link puede registrarse como desarrollador y encontrar una documentación apropiada para consumir el API. El buscador sólo debe mostrar un máximo de 10 resultados a la vez, cuando se seleccione el personaje deseado se debe desplegar la lista de comics asociados a él, de los cuales se debe poder visualizar el detalle y también se debe poder guardar en mi lista de comics favoritos.
Con respecto a la lista de cómics favoritos, no puede haber cómics repetidos en dicha lista. Además, se requiere un mecanismo para añadir tres cómics al azar, estos tres cómics tienen que ser seleccionados de los que se encuentran asociados a la lista de Personajes. Por último, la lista de cómics debe ser persistida en localStorage, para que no se pierda en caso que el usuario cierre la SPA y se pueda visualizar cuando regrese a ella.

Una vez haya terminado el código describa en un documento brevemente:
1. Las capas de la aplicación (por ejemplo: capa de persistencia, vista, de aplicación, etc) y qué clases u objetos pertenecen a cuál.
2. La responsabilidad de cada clase u objeto creada.

Recibirá puntos extra si:
1. Usa git o SVN y la historia del programa muestra su progreso en el desarrollo, usando commits con unidades de funcionalidad (5 puntos)
2. Usa pruebas unitarias y/o de validación (acceptance) que demuestran el buen funcionamiento del programa (15 ptos)

Al completar el challenge, por favor enviarnos la url con la aplicación funcional, el código que lo soluciona o el link al repositorio donde se encuentra el código, el documento con la breve descripción de su solución.

2. CODE REFACTORING (20 puntos)
Refactorice y envíe el código y en un documento explique:
1. Las malas prácticas de programación que en su criterio son evidenciadas en el código
2. Cómo su refactorización supera las malas prácticas

```
function post_confirm(params) {
const id = params.service_id;
let servicio = Service.find(id);
// console.log(servicio);
if (servicio != NULL) {
if (servicio.status_id == '6') {
```

```
return {error: '2'};
}
if (servicio.driver_id == NULL && servicio.status_id == '1') {
servicio = Service.update(id, {
driver_id: params.driver_id,
status_id: '2'
//Up carro
//, pwd: md5(params.pwd)
});
Driver.update(params.driver_id, {
available: '0'
});
driverTmp = Driver.find(params.driver_id);
Service.update(id, {
car_id: driverTmp.car_id
//Up carro
//, pwd: md5(params.pwd)
});
//Notificar a usuario!!
var pushMessage = "Tu servicio ha sido confirmado!";
/* servicio = Service.find(id);
push = Push.make();
if (servicio.user.type == '1') {//iPhone
pushAns = push.ios(servicio.user.uuid, pushMessage);
} else {
pushAns = push.android2(servicio.user.uuid, pushMessage);
} */
servicio = Service.find(id);
push = Push.make();
if (servicio.user.uuid == '') {
return {error: '0'};
}
if (servicio.user.type == '1') {//iPhone
push.ios(servicio.user.uuid, pushMessage, 1, 'honk.wav', 'Open', {service_id: service.id});
} else {
push.android2(servicio.user.uuid, pushMessage, 1, 'default', 'Open', {service_id: service.id});
}
return {error: '0'};
} else {
return {error: '1'};
}
} else {
return {error: '3'};
```

3. PREGUNTAS (10 puntos)

Responda las siguientes preguntas

1. ¿En qué consiste el principio de responsabilidad única? ¿Cuál es su propósito?
2. ¿Qué características tiene según su opinión “buen” código o código limpio?