LogServiceTest:

Caso de Test	Problema	Solución
addLogs	El número de logs que devuelve es el doble que el esperado.	Eliminar un list.add(log) que se encuentra dentro del método addLog en la clase LogService ya que agrega el log dos veces cada vez que se llama al método.
hasAllLogsConcurrent	El número de eventos que devuelve es el doble que el esperado.	Eliminar un list.add(log) que se encuentra dentro del método addLog en la clase LogService ya que agrega el log dos veces cada vez que se llama al método.
getLogsByUser	Se crea dos instancias de un usuario con el mismo nombre, por lo tanto deberían ser iguales, pero cuando se consulta el por los logs de ese usuario devuelve 1 en vez de 2 porque interpreta a los usuarios como 2 diferentes.	Implementar el método equals y hashcode de la clase User utilizando el nombre de usuario para que haga la comparación de la forma que se desea.

TodoServiceTest:

Caso de Test	Problema	Solución
testTimerDelete	Borra mas ToDo de los esperados.	Hay un error en la lógica de comparación entre los segundos pasados para eliminar los ToDo y los segundos que pasaron hasta el momento. Se debe cambiar el segundosDelTodo < sec , que se encuentra dentro del método deleteOldMessages de la clase TodoService, por segundosDelTodo > sec para que la lógica sea la correcta.
testCreateTodo	Se crea un ToDo con determinado contenido, al obtenerlo y comparar el contenido del mismo ToDo guardado con el contenido con el que se creó arroja que el contenido no es el esperado.	En el método createNewTodo de la clase TodoService no se seteaba el contenido a la hora de crear el todo. Para solucionarlo basta con agregar la línea nuevo.setContent(content); antes de hacer el

		addTodo(nuevo) para setear el contenido correctamente.

CurrentUserTest:

Caso de Test	Problema	Solución
testGetsDefaultUser	La llamada al método getCurrent de CurrentUserService no devuelve el usuario por defecto, devuelve null.	Hay un error en la lógica del método getCurrent de CurrentUserService. Si current != null devuelve el usuario por defecto, lo que esta mal, porque si current == null debería devolver el usuario por defecto. Se cambio current != null por current == null para solucionar el error.
testCurrentConcurrent	Hay un error de concurrencia entre los 2 threads.	Se agrego el modificador synchronized al método getCurrent de CurrentUserService para que el método este sincronizado, es decir, que solamente un subproceso puede acceder a dicho método a la vez.

UserServiceTest:

Caso de Test	Problema	Solución
testLoginUser	No devuelve el Id esperado.	En el método login de la clase UserService se cambió lastId = lastId++ por lastId = ++lastId. Esto se debe que al hacer un postincremento primero se asigna el valor a la variable y luego se aumenta el valor, lo que provocaba que los Ids no sean los correspondientes.
testAddUsersAndClear	Produce un exceso de memoria ocupada por no limpiar los logs al ejecutar el método clearUsers de UserService.	Se agrego la línea this.logSvc.clear(); en el método clearUsers de la clase UserService para que se limpien también los logs cuando se limpien los usuarios.

testAddNullUser	No se contempla el caso de agregar un	Se agrego
		`if( user == null )`
		`    return null;`
		al principio del método addUser de la clase UserService para que devuelva un null al agregar un null y no arroje un error.

	null en el método addUser de la clase UserService.	
testSortedByName	El método getUsers de la clase UserService no los devuelve ordenados por nombre si no por Id.	Cambiar la línea `result.sort(Comparator.comparing(User::getId));` por la línea `result.sort(Comparator.comparing(User::getName));` del método getUsers para que los ordene por nombre en vez del Id.