Andrés Santiago Triana

Juan Sebastián Ramírez

Iteración 3 Vacuandes

Resultados logrados y no logrados

Proceso

Nuestra ejecución en esta iteración fue simple, esta vez queríamos priorizar que todos los puntos funcionaran, tanto los de funcionales como los de consulta. Lo que hicimos a lo largo de estas semanas fue trabajar de manera constante de punto en punto. En primer lugar, desarrollábamos la interfaz, luego de ahí partíamos a vacuandes en la clase negocio, luego proseguíamos con la clase persistencia y finalmente modificábamos las clases que mandaban sentencias SQL. Una vez que todos esos procesos eran completados, los probábamos de manera manual y corregíamos cualquier error que surgiera. Posteriormente, hicimos el mismo proceso con los requerimientos de consulta. Finalmente, hicimos las pruebas de los requerimientos.

Resultados

Todos los requerimientos, tanto funcionales como de consulta y no funcionales se ejecutan de manera correcta y sobre todo coherente. **Todas las pruebas de todos los métodos están aprobadas**.

RF1 a RF7

Todos los requerimientos de la iteración anterior, incluidos los que antes no servían, fueron arreglados y quedaron listos. Verificamos cada uno de los métodos primero de manera manual y luego con tests.

RF8

Hicimos mucho énfasis en esta iteración en priorizar que el software se pareciera lo más posible al manejo de las vacunas en la vida real.

En este caso, nuestro manejo del proceso fue el siguiente, en primer lugar, para hacer el proceso de registrar llegada de lote de vacunas a la oficina regional EPS, lo que hicimos fue que aumentara el valor de cantidad de vacunas actuales que tenía guardadas y además que también creara esas nuevas vacunas que llegan a la oficina. Sin embargo, para poder agregarlas debe hacer una verificación que revisa que haya suficiente espacio para almacenar las vacunas, para esto usamos la comparación:

(cantidad vacunas que llegan + cantidad vacunas actuales) <= cantidad de vacunas en esa oficina

De esta manera nunca se van a poder agregar más vacunas que las que se pueden guardar por la capacidad de la oficina.

RF9

Para el manejo de la llegada de un lote de vacunas a un punto de vacunación hicimos un código un poco más detallado, ya que lo que hacemos es simular el “transporte” de las vacunas de la oficina regional al punto de vacunación. Por lo tanto, cuando se registra la llegada al punto de vacunación, lo que sucede es que la cantidad de vacunas en la oficina regional se le resta la cantidad de vacunas que llegan al punto de vacunación. De esta manera es coherente en todo momento el número de vacunas que hay y en qué lugar están. Adicionalmente, se verifica que no se exceda la capacidad del punto de vacunación y también que no salgan más vacunas del punto de vacunación que las que están guardadas actualmente. Todo ello se hace con las siguientes comparaciones:

(cantidad vacunas que van a salir de la oficina) <= cantidad de vacunas actuales en la oficina

(cantidad vacunas que llegan + cantidad vacunas actuales) <= cantidad de vacunas en ese punto

RF10

Asignar un punto de vacunación a un ciudadano es una operación que no tiene muchos problemas, esto se debe a que siempre se puede asignar un ciudadano a un punto de vacunación, esto se debe a dos factores: en primer lugar, a un punto de vacunación se le pueden asignar un número ilimitado de ciudadanos debido a que la verificación de capacidad es una de atención, es decir, solo nos preocupa que en las citas no se sobrepase el limite de ciudadanos, ya que el personal solo va a tener complicaciones para vacunar si se pasa el limite de ciudadanos que pueden atender en un rango de media hora. En segundo, lugar, desafortunadamente no le pusimos condiciones al punto de vacunación debido a limites de tiempo que teníamos, es un proyecto bastante largo y hacer todos esos cambios en tantas capas nos iba a quitar mucho tiempo para ser solo una pequeña parte de un requerimiento funcional.

RF11

En este requerimiento ya habíamos antes tomado en cuenta la capacidad de atención de los puntos de vacunación, por lo cual no tuvimos problemas dejarlo listo cumpliendo las restricciones de capacidad. Nuestra manera de verificarlo es: primero verificamos en la tabla de citas la cantidad de citas que hay para la fecha, hora y punto en el que se quiere poner la cita, de esta manera sabemos con precisión cuantas personas van a tener una cita al mismo tiempo en ese mismo lugar. Posteriormente, usamos la siguiente comparación para hacer la verificación:

(citas actuales + 1) <= capacidad de atención simultanea del punto

De esta manera, se evita que haya más citas de las que puede soportar un punto en un intervalo de media hora, que es lo que tarda una cita siempre.

Adicionalmente, se hace una verificación de que haya vacunas disponibles, para ello lo que hacemos es con un método, traemos una vacuna al azar registrada al punto de vacunación, bajo la condición de que la vacuna no esté marcada como usada, las vacunas que están marcadas de esta

manera son porque ya tienen asignadas una cita o ya sucedió la cita. Así garantizamos que no se pueda asignar una cita si el punto no tiene vacunas disponibles y que no se le asignen al ciudadano vacunas que ya fueron o van a ser utilizadas.

RF12

Para este requerimiento funcional ya teníamos un método muy bien implementado que permitía con facilidad que el estado de vacunación se actualice, esto ya lo habíamos preparado antes y por esto funciona bien hasta ese punto. El problema aquí fue de tiempo nuevamente, esto se debe a que no pudimos hacer el registro de inventario y garantizar que las vacunas fueran iguales.

RF13

Este requerimiento en realidad quedó resuelto con los métodos de los RF14 Y RF15, debido a que un cambio de estado, para nosotros, era equivalente a habilitar y deshabilitar un punto de vacunación. Lo que si hay que mencionar en este punto es que tuvimos que modificar la estructura de punto para que ahora tuviera el parámetro de habilitado o deshabilitado y que cuando se crea un nuevo punto, por defecto se crea habilitado.

RF14

El requerimiento funcional 14 fue un poco difícil de hacer, se creó un método en la persistencia que hacía varios pasos en los distintos SQL que estaban involucrados. En primer lugar, se cambiaba el booleano de habilitado a deshabilitado en el punto de vacunación. En segundo lugar, todos los ciudadanos que estaban inscritos en ese punto de vacunación eran cambiados a otro punto dado por parámetro. En tercer lugar, se guardaba una lista de las citas eliminadas, puesto que tocaba eliminar todas las citas del punto que cumplieran la condición de ser mayores o iguales a la fecha actual. Una vez guardada la lista, se eliminaban las citas. Finalmente, la parte más difícil, asignar automáticamente nuevas citas en el nuevo punto de vacunación dado por parámetro. Para ello lo que hicimos fue tomar todas las citas e intentar inscribirlas en la misma fecha y la misma hora en el nuevo punto, sin embargo, si se verificaba que no se podía porque sobrepasaba la capacidad del punto de vacunación en ese horario, entonces lo que se hacía es que se cambiaba la hora a media hora después y se volvía a intentar agregar. Si la hora llega a sobre pasar la hora a la que cierran los puntos, entonces lo que se hace es que el día se aumenta en 1 y la hora cambia a la primera hora disponible. Y también, si se está en el día viernes y a la ultima hora, entonces se cambia al día lunes de la siguiente semana la fecha y la hora se reinicia a la primera hora disponible en el punto.

RF15

Este requerimiento funcional era más sencillo que deshabilitar, puesto que lo que se hace es que simplemente se cambia el booleano de habilitado y nuevamente se pueden volver asignar citas en el punto.

RFC1 A RFC5

Todos esos requerimientos de consulta funcionan correctamente, excepto por el de consulta 3, fue un error por nuestra parte y no nos dimos cuenta de que no lo habíamos acabado sino hasta muy tarde en la iteración. El funcionamiento de nuestros métodos de consulta es que la interfaz pide los Strings a vacuandes y vacuandes a su vez lo pide de la persistencia, la cual está encargada de generar los Strings según lo que se le pida.

RFC6

No vimos la necesidad de agregar nuevos métodos de consulta, pues los que teníamos hasta ese punto estaban muy bien hechos y proveían mucha información relevante.

RFC7

Este requerimiento funciona correctamente, a pesar de haberse presentado inicialmente un problema que se pudo solucionar. El problema principal inicialmente, al ver este requerimiento, era la complejidad, esto se debe a que no importaba el rango que se diera, si era por fechas, por horas, por días o por semanas, igualmente era necesario recorrer todas las citas en esos rangos de todas las horas de todos los días que estuvieran disponibles, lo cual era demasiado, pues por ejemplo, para 1 semana, en un solo punto, habían disponibles 140 espacios de cita y si el punto tenía un límite de capacidad de atención simultanea de 10 personas, entonces había que verificar 1400 citas para poder decir si el punto estaba o no lleno o vacío. La complejidad temporal hubiese sido muy alta si se hubiera optado por hacer el método tal y como se planteó inicialmente. Es por esto que se hizo una modificación, lo primero que se hace es pedir una tabla que trae todas las citas que existen en el rango de tiempo especificado, esta tabla llega a la persistencia como una lista de tipo objeto, posteriormente, con esa lista de tipo objeto se toman los parámetros que se piden nuevamente para poder traer solo los puntos que cumplen la condición de tener citas registradas.

RFC8

El requerimiento funcional 8 era particularmente complejo, esto se debía a que para poder tener cohortes flexibles, era necesario modificar la función SQL con cada uno de los casos, por lo tanto, se hubiesen necesitado demasiados métodos para poder cumplir cada uno de los casos que

teníamos. La solución que se hizo fue crear con concatenadores la sentencia de SQL que los parámetros de busqueda serían procesados independientemente por distintos métodos, estos métodos lo que hacen es que crean la sentencia que va a ser concatenada con la sentencia general. Por ejemplo, si ser recibe un arrayList de puntos en los que se quiere verificar la característica, el método lo que hace es que crea una sentencia “WHERE punto_vacunacion = ? OR punto_vacunacion = ? ”, se debe notar que la sentencia por si sola es capaz de poner solamente el WHERE inicial si solo recibe un parámetro, también de no poner nada si no recibe ninguno y de poner los OR respectivos si recibe más de un parámetro. Luego, todas las partes de las sentencias se concatenaban a la que se iba a enviar y luego a partir de esta se formaba el String de respuesta.

RFC9

Para este requerimiento podía suceder un problema similar al del RFC7 donde tocaba verificar demasiadas citas para encontrar los ciudadanos que estuvieron en contacto. Es por esto que, en primer lugar, se usó la tabla de citas para no verificar horarios vacíos y en segundo lugar, se verifico que la cantidad de citas en un rango de fecha, hora y punto fuera diferente de 1, pues si solo hay un ciudadano no hay probabilidad de que se hubiera cruzado con nadie más. El método imprime los puntos en los que hubo cruces, en qué momento hubo y también la información de todos los ciudadanos que se cruzaron en esos puntos.

Balance de pruebas

Se hizo un total de 23 archivos JUnit test para cada uno de los requerimientos funcionales. Todas las pruebas de todos los requerimientos fueron aprobados, sin embargo, la prueba para el método de RFC3 no existe debido a que ese punto no fue implementado. A continuación, se adjunta una imagen del Excel con la información detallada de cada una de las pruebas.

Requerimiento	Nombre	Input	Output	Estado
RF1	verificarReq1Registrar	Condicion que se desea modificar	Condicion modificada	Correcto
RF2	verificarReq2RegistrarEstado	Estado de vacunacion a registrar	Estado registrado	Correcto
RF3	verificarReq3Registrar	Oficina regional a registrar	Oficina regional registrada	Correcto
RF4	verificarReq4RegistrarUnUsuario	Ciudadano de prueba, usuario asociado con el ciudadano de prueba.	usuario y ciudadano registrados	Correcto
RF5	verificarReq5Registrar	Ciudadano a registrar	Ciudadano registrado	Correcto
RF6	verificarReq6RegistrarUnPuntoVacunacion	Punto de vacunacion de prueba a registrar.	Punto de vacunacion registrado	Correcto
RF7	verificarReq7Registrar	Trabajador a registrar	trabajador registrado.	Correcto
RF8	verificarReq8Registrar	Oficina regional a registrar junto a la cantidad de vacunas, con la condicion de prueba que se quiere añadir	Oficina regional de prueba registrada, cantidad de vacunas creadas en la bd y registradas correctamente en la oficina	Correcto
RF9	verificarReq9Registrar	Oficina regional, punto de vacunacion y cantidad de vacunas que entraran al punto de vacunacion de prueba	Punto de vacunacion registrado y vacunas asiganadas correctamente	Correcto
RF10	verificarReq10Registrar	Punto de vacunacion de prueba para asingar al ciudadano y ciudadano de prueba para ser asigando.	Ciduadano registrado y asignado correctamente al punto	Correcto
RF11	verificarReq11Asignar	Ciudadano a registrar y cita a registrar para el ciudadano	Ciudadano registrado y cita correctamente asignada	Correcto
RFC1	verificarReqC1TodosRegistros	id punto de vacunacion	todos los ciudadanos atendidos por el punto de vacunacion dado por parametro	Correcto
RFC1	verificarReqC1FechaEspecifica	id punto de vacunacion y fecha especifica para la busqueda	todos los ciudadanos atendidos por el punto de vacunacion dado por parametro en la fecha dada	Correcto
RFC1	verificarReqC1RangoFechas	id punto de vacunacion y el rango de las fechas para la busqueda	todos los ciudadanos atendidos por el punto de vacunacion dado por parametro dentro de las fechas dadas por parametro	Correcto
RFC1	verificarReqC1RangoHora	id punto de vacunacion y rango de fechas para la busqeuda	todos los ciudadanos atendidos por el punto de vacunacion dado por parametro dentro de las horas dadas por parametro	Correcto
RFC2	verificarReqC2TodosRegistros	nada	Los 20 puntos mas efectivos de toda la bd	Correcto
RFC2	verificarReqC2FechaEspecifica	Fecha especifica para la busqueda	los 20 puntos mas efectivos de toda la bd en la fecha dada	Correcto
RFC2	verificarReqC2RangoFechas	Rango de fechas para la busqeuda	los 20 puntos mas efectivos en el rango de fechas dado	Correcto
RFC2	verificarReqC2RangoHora	rango de horas para la busqueda	los 20 puntos mas efectivos en el rango de hora dado	Correcto
RFC7	verificarReqC7RangoHorasSobrecupo	tipo punto vacunacion y rango de horas para la busqeuda	resultado de la busqeuda de sobrecupo	Correcto
RFC7	verificarReqC7RangoFechasSobrecupo	tipo punto vacunacion y rango de fechas para la busqueda	resultado de la busqeuda de sobrecupo	Correcto
RFC7	verificarReqC7FechaEspecificaSobrecupo	tipo punto vacunacion y fecha especifica	resultado de la busqeuda de sobrecupo	Correcto
RFC7	verificarReqC7RangoHorasFaltaCupo	tipo punto vacunacion y rango de horas para la busqueda	resultado de la busqeuda falta cupo	Correcto
RFC7	verificarReqC7RangoFechasFaltaCupo	tipo punto vacunacion y rango de fechas para la busqeuda	resultado de la busqeuda falta cupo	Correcto
RFC7	verificarReqC7FechaEspecificaFaltacupo	tipo punto vacunacion y fecha especifica	resultado de la busqeuda falta cupo	Correcto

Imagen 1. Tabla de pruebas en Excel.

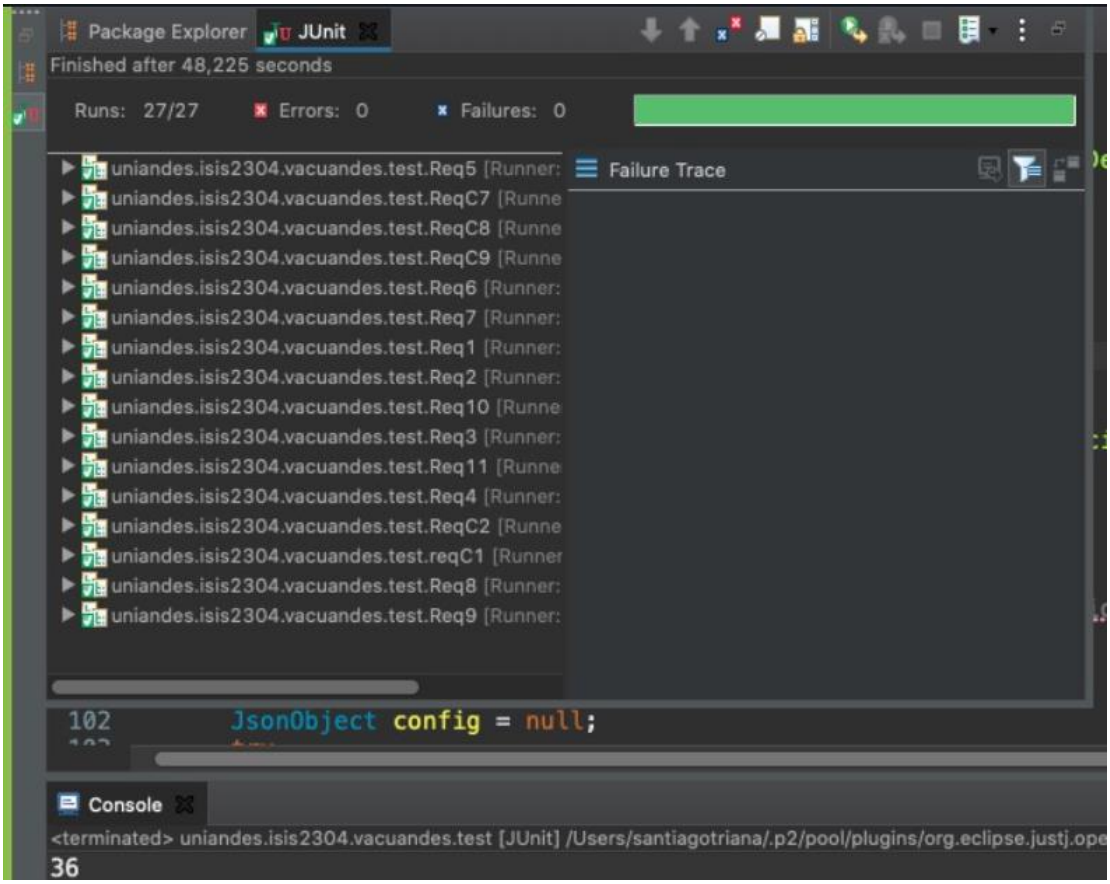

Imagen 2. Imagen de todas las pruebas corriendo en Java.