

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

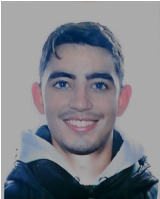
DP2-Informe de Análisis Student 5 D3



Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas II

Curso 2023 – 2024

<u>Group:</u>	C1.016
<u>Repository:</u>	https://github.com/luchersol/Acme-SF-D03
<u>Student #5</u>	
	
UVUS:	dangalcan
Name:	Galván Cancio, Daniel
Email:	dangalcan@alum.us.es
<u>Date:</u>	Sevilla Abril 23, 2024

Índice de contenido

1. Versiones	2
2. Lista de los registros de análisis	2
3. Conclusión del análisis	2
4. Enlace de la validación del profesor	2
5. Bibliografía	2

1. Versiones

Versión	Fecha	Autor
1.0	13/03/2024	Daniel Galván Cancio
1.1	22/04/2024	Daniel Galván Cancio
1.2	23/04/2024	Daniel Galván Cancio

2. Lista de los registros de análisis

-Task 6: Operations by **auditors** on **code audits**:

- List the **code audits** that they have created.
- Show the details of their **code audits**.
- Create, update, or delete their **code audits**. **Code audits** can be updated or deleted as long as they have not been published. For a **code audit** to be published, the mark must be, at least, "C".

- Task 7: Operations by **auditors** on **audit records**:

- List the **audit records** in their **code audits**.
- Show the details of their **audit records**.
- Create and publish an **audit record**.
- Update or delete an **audit record** as long as it is not published.

- Task 8: Operations by **auditors** on **auditor** dashboards:

- Show their **auditor** dashboards.

- Task 17: Operations by anonymous principals on user accounts:

- Sign up to the system and become an **auditor**.

- Task 18: Operations by **auditors** on user accounts:

- Update their profiles.

-Task 19: Operations by any principals on **code audits**:

- List the **code audits** in the system that are published.
- Show the details of the **code audits** that they can list (including their **audit records**).

- Task 20: Produce an analysis report.

- Task 21: Produce a planning and progress report.

- Task 22: Produce a lint report.

3. Conclusión del análisis

- Task 6: Para esta tarea desempeñé el rol de desarrollador. Realmente la complicación estuvo en entender el método `unbind`, y en asegurarme de que no se puedan vulnerar las reglas de negocio. Como fue la primera tarea que hice, estaba un poco perdido pero no hubo mucho problema. El frontend me resultó más sencillo de lo que pensé, ya que a priori utilizar `jsp`, que es una tecnología totalmente nueva para mí, pensé que sería todo un reto. Es cierto que no es algo trivial, pero el framework hace que sea sencillo hacer las vistas. He programado el requisito de manera que los auditores puedan listar sus auditorías (las que han creado ellos). Podrán verlas estén o no en modo borrador. Además de esto, al publicar un `codeAudit` al inicié decidí que los `auditRecords` que estén en modo borrador, es decir, que no se hayan publicado, se eliminen automáticamente. Tomé esta decisión porque para que una auditoría se pueda publicar debe tener una nota de C, y una vez publicado no puede modificarse, lo cual implica que tampoco se podrán publicar las `auditRecords` que pertenecen a ellas. Por tanto, una vez se cumplía la condición de tener la nota suficiente, se podía publicar. Al haberse publicado, como nunca se podrían publicar los `auditRecords` que no estén publicados, y nunca contribuirían en las estadísticas ni en la nota, era mejor que se eliminen. Como el usuario puede ver en todo momento cuáles están publicadas y cuáles no, y tiene que darle a 2 botones para publicar el `codeAudit`, veía lógica esa forma de proceder. Además, lo consulté en el feedback del 22/04/2024 con el profesor Manuel Jesús y me dijo que si lo argumentaba de esa forma era una estrategia de desarrollo de la historia de usuario válida. No obstante, en el foro de la asignatura una persona comentó esto a Rafael Corchuelo y como le indicó que un `codeAudit` NO podía publicarse hasta que todos sus `auditRecords` estuvieran publicados, pues tuve que cambiar esa feature. Realmente no fue muy difícil, pero sí que es cierto que tuve que hacer los cambios cerca de la última hora.

- Task 7: Para esta tarea desempeñé el rol de desarrollador. Tuve que añadir un atributo nuevo a AuditRecord, y por tanto al diagrama UML. Se trata del atributo draftMode, que es el responsable de indicar si algo está publicado o no. Realizar la parte del backend fue sencilla menos por lo de recalcular la nota cada vez que se publica un auditRecord. También tuve problemas al conectar en el frontend la parte de los auditRecords con los codeAudits, porque no sabía si había que hacer un listado dentro de los detalles de los codeAudits, si había que hacer una vista aparte. Al final resultó que tenía que poner un botón en la vista de detalles del codeAudit para que me lleve a la lista de auditRecords asociados a ese codeAudit. Me hice un lío con el masterId y el id en el frontend pero al final pude resolver la confusión y hacer bien la tarea.

- Task 8: Para esta tarea desempeñé el rol de desarrollador. Con respecto al dashBoard en lo que más problemas tuve fue a la hora de hacer la desviación típica, ya que por algún motivo no me compilaban las queries. Por lo demás fue todo relativamente sencillo de implementar. Al final solventé la no compilación de la desviación típica, y pude ver que a la hora de calcular el periodo tenía problemas con las unidades. Por cómo se trataban los timeStamps en JPA vs por cómo se trataban en SQL obtenía números sin sentido (en lugar de 1.5h obtenía 310, en lugar de 1h obtenía 10000). Por tanto, y tras no poder dividir entre 3600 o entre 3600000 para solventarlo, opté por usar streams para poder hacer las operaciones relativas a las estadísticas del periodo. El problema radicaba en que JPQL no soporta poner "startDate.getTime()" y "endDate.getTime()" en la query, y sin hacer eso no podía llegarse a una unidad de medida del tiempo común y por eso saltaban esos números tan extraños. He de decir que las estadísticas están hechas con respecto a las codeAudits que están publicadas, y tomando los auditRecords de esas codeAudits que también están publicados, siendo todos esos objetos del auditor con respecto al que se calculan las estadísticas. La medida del periodo de los auditRecord he decidido que sean las horas, y, además, para la media, las estadísticas de los auditRecords y la desviación típica, he tomado las estadísticas sin hacer una diferenciación entre los tipos de codeAudit, ya que creo que es mejor así.

- Task 17: Para esta tarea desempeñé el rol de desarrollador. Fue bastante más sencilla de lo que pensaba porque en acme jobs teníamos un ejemplo que ilustraba muy bien como había que hacerse. No tuve dificultad ninguna.

- Task 18: Para esta tarea desempeñé el rol de desarrollador. Aquí me ocurrió un poco como en la tarea anterior, fue bastante rápido de implementar y más fácil de lo que pensaba.

- Task 19: Para esta tarea desempeñé el rol de desarrollador. Como ya tenía hechos los requisitos 6 y 7, fue prácticamente copiar y pegar pero cambiando ciertos privilegios y eliminando la parte de modificar y añadir datos a la base de datos. No tuve ningún problema. He hecho que se puedan ver solo las codeAudits que estén publicadas, junto con sus auditRecords, siendo estos siempre los publicados.

- Task 20: Para esta tarea desempeñé el rol de project manager. En reporte de análisis fue fácil de hacer en el sentido de que ya tenía la experiencia de las entregas anteriores, por lo que no tuve dudas, pero fue más tardado porque he tenido que esperar a tener todas las tareas hechas para poder hacerlo. Lo único nuevo ha sido añadir la validación del profesor.

- Task 21: Para esta tarea desempeñé el rol de project manager. Al igual que con la 20, me pasó lo mismo, tuve que esperar a tener todas las tareas hechas para hacerlo, pero con la experiencia de las otras 2 entregas no fue un problema realizarlo.

- Task 22: Para esta tarea desempeñé el rol de project manager. Esta sí me costó más porque no estaba acostumbrado a cómo se hacía y también porque nunca antes había hecho un linting report. No obstante, al final puedo decir que más o menos estoy conforme con mis resultados, y espero que estén acorde con lo exigido en los requisitos.

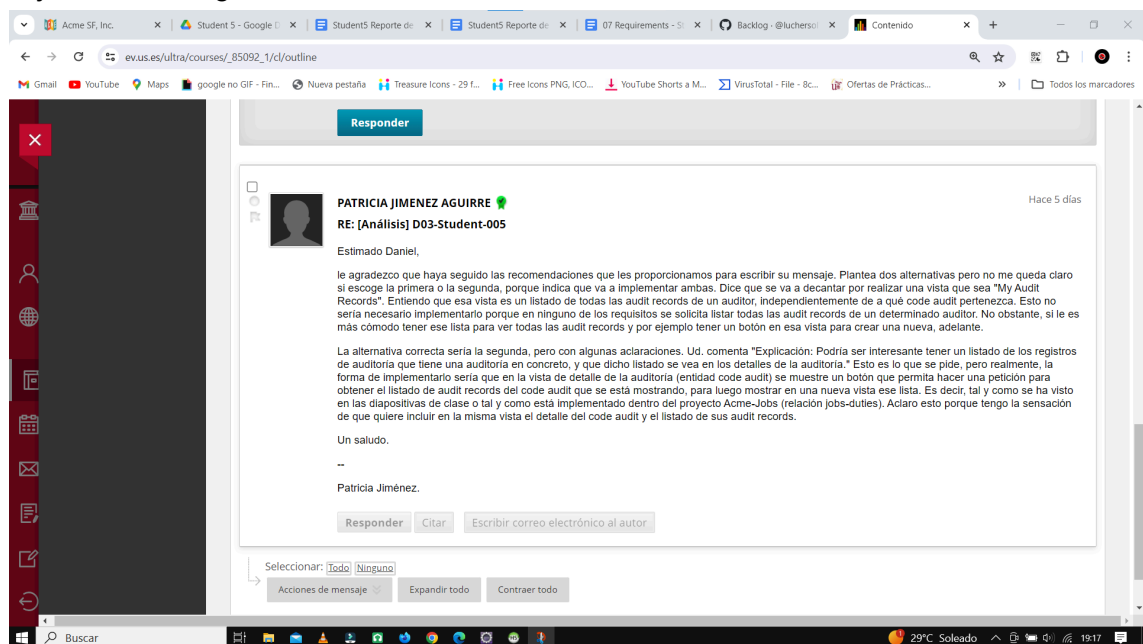
4. Enlace de la validación del profesor

Para esta entrega realicé un registro. Tenía una duda sobre cómo se deberían representar los audit-records dentro del frontend. Al inicio pensé que debería haber una vista donde aparecieran todos los audit-records. No obstante, como se indicó que no era necesario opté por no hacerlo. En la pregunta que puse en el foro, también pregunté si debían listarse los auditRecords de un codeAudit en la misma vista en la que se muestran los detalles del codeAudit al que pertenecen. Esta opción también fue descartada. La que quedó como correcta y, como la que debía tomarse fue la de añadir un botón que llevara a otra vista con todos los auditRecords del codeAudit deseado. Dicho botón se encontraría en la vista de show codeAudit, es decir, en la de detalles del codeAudit.

Enlace a validación del profesor/a:

https://ev.us.es/ultra/courses/_85092_1/cl/outline

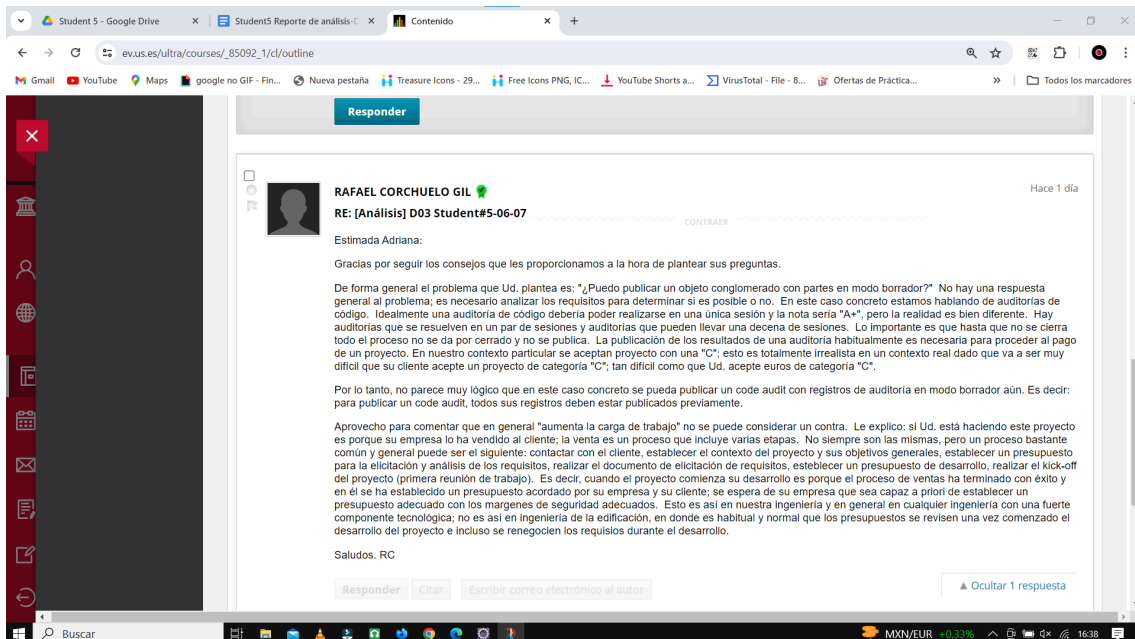
En caso de que no pueda visualizarse la consulta con la respuesta del profesor, adjunto una imagen de la misma:



Sobre lo mencionado sobre que los codeAudit no podrían publicarse hasta tener todos sus auditRecords publicados, está aquí:

Enlace a validación del profesor/a: https://ev.us.es/ultra/courses/_85092_1/cl/outline

En caso de que no pueda visualizarse la consulta con la respuesta del profesor, adjunto una imagen de la misma:



5. Bibliografía

No hay bibliografía presente para esta entrega.