Universidad de Sevilla  
Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática – Ingeniería del Software  
Diseño y Pruebas 2

Curso 2023 – 2024

**Reporte de análisis estudiante 4**

**Nombre completo**

**correo**

**Repositorio:** [https://github.com/DP2-C1-013/Acme-SF-D01-24.1.0](https://github.com/DP2-C1-013/Acme-SF)

| **Fecha** | **Versión** |
| --- | --- |
| 14/2/2024 | v1.0 |

[**1. Resumen 2**](#_4idpezggeose)

[**2. Control de versiones 3**](#_kuytbm87rdrm)

[**3. Introducción 3**](#_7js13dvc08tx)

[**4. Contenidos 4**](#_r1vl5b5325v9)

[4.1. Entregable 1: Modificación del menú de anónimos 4](#_ciwqr5jxqhwz)

[4.2. Entregable 2 4](#_wiv75vxbxnk0)

[4.2.1 Entidad Code Audit 5](#_qkmeb4jji19v)

[4.2.2 Entidad Code Record 5](#_jqcfyr3te8pz)

[4.2.3 Formulario Auditor Dashboard 5](#_dg55qyxmagxf)

[4.2.4 Rol Auditor 6](#_evkx8wqkb8fu)

[4.2.5 Datos de muestra 6](#_miaxpu50wyvg)

[**5. Conclusión 6**](#_phy2ya37znhj)

[**6. Bibliografía 6**](#_78o4gtq9n3uv)

# 1. Resumen

Este reporte tiene como objetivo realizar un listado del análisis de los distintos requisitos individuales asignados al estudiante #5 para el proyecto de la signatura. Para cada requisito se dará información sobre las conclusiones a las que ha llegado el equipo y de las decisiones tomadas.

# 

# 2. Control de versiones

| Nº de revisión | Fecha | Descripción |
| --- | --- | --- |
| 1 | 14/02/2024 | Desarrollo inicial del reporte de análisis |
| 2 | 08/03/2024 | Análisis de los requisitos de la segunda entrega |

# 

# 3. Introducción

El reporte de análisis está centrado en el análisis de los requisitos asignados al estudiante 5. En esta versión del documento, correspondiente al entregable 2 del proyecto, la descripción del análisis llevado a cabo se centra principalmente en explicar qué cambios se han hecho a grandes rangos, y el por qué de ciertas decisiones de diseño; cómo relacionar entidades, qué tipo de dato usar para cada atributo en concreto…

El contenido del reporte de análisis está estructurado en los siguientes puntos:

1. Resumen ejecutivo: resumen muy general del contenido principal del reporte.
2. Control de versiones del documento.
3. Introducción: a diferencia del resumen ejecutivo, se realiza un resumen más detallado del contenido principal al mismo y se explica la estructura del documento.
4. Contenidos: punto principal del reporte en el que se entra con todo detalle en el análisis de los distintos requisitos.
5. Conclusión.
6. Bibliografía.

# 4. Contenidos

## 4.1. Entregable 1: Modificación del menú de anónimos

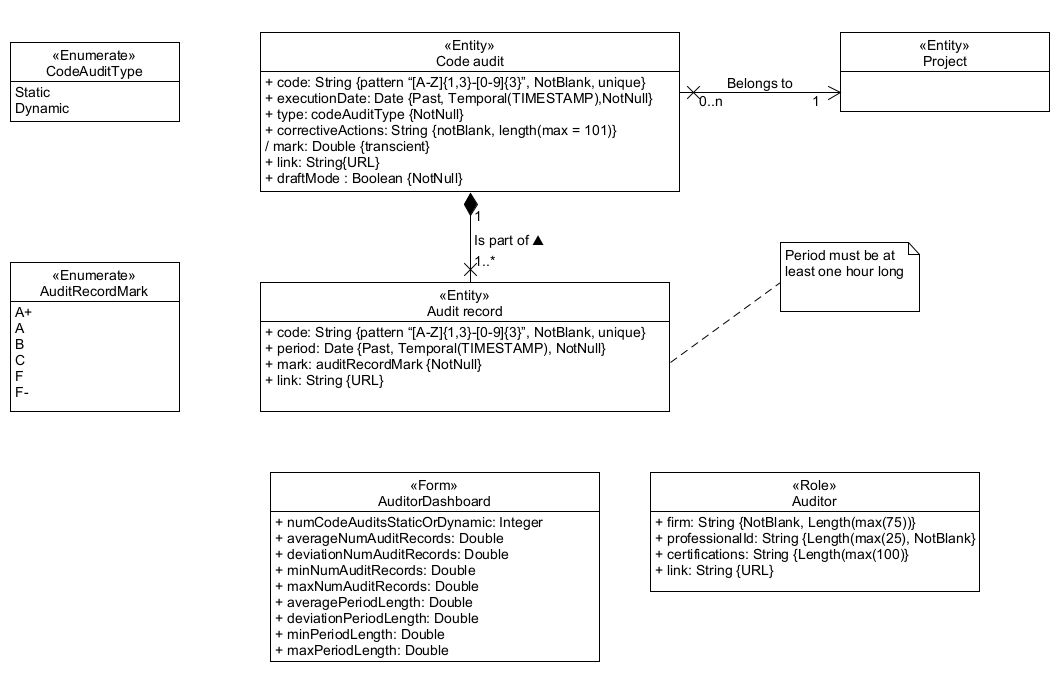
*Modify the anonymous menu so that it shows an option that takes the browser to the home page of your favourite web site. The title must read as follows: “〈id-number〉: 〈surname〉, 〈name〉”, where “〈id-number〉” denotes your DNI, NIE, or passport number, “〈surname〉” denotes your surname/s, and “〈name〉” denotes your name/s.*

Este requisito conlleva incluir de forma individual un enlace a la página web preferida de cada desarrollador. Además, será necesario que en lugar del texto del enlace, en la vista aparezca los datos del desarrollador siguiendo el patrón de DNI-Apellidos,Nombre. Para realizar dichas modificaciones será necesario añadir una entrada en la sección *anonymous* del fragmento *menu.jsp* con el enlace y en el que se muestre el título especificado en los archivos *menu-es.i18n* y *menu-en.i18n* (se tiene que poner la información en ambos ficheros para su correcto funcionamiento tanto en español como inglés).

Como enlace he decidido usar <https://www.freecodecamp.org/>.

## 4.2. Entregable 2

En este entregable con el objetivo de cumplir con los requisitos para el Student#5 se ha creado una serie de entidades relacionadas entre sí, un Form, un Rol y datos de muestra en forma de csv para cada entidad, viendo en cada caso cómo persisten en la base de datos.



Como se puede ver en el UML, se ha considerado que la relación entre las entidades Code Audit y Code Record sea un conglomerado, más concretamente una composición, pues no tiene sentido que existan Code Records que no pertenezcan a ningún Code Audit o que pertenezcan a varios.

### **4.2.1 Entidad Code Audit**

Estas “audiciones de código” sirven para garantizar la calidad de los proyectos, por lo que la entidad que guarda su información está relacionada con la entidad Project. Esto se ha traducido a nivel de código en un Many to one.

Adicionalmente, como ya se ha explicado, esta entidad forma un conglomerado con Code Record.

De todos sus atributos, cabe destacar la siguiente decisión de diseño:

Nota: la nota de una inspección de código puede ser A+, A, B, C, F, F-. Su valor va a depender de la moda de otro atributo “nota” de los Code Record que contiene cada inspección. Se ha implementado usando un enumerado, aunque también se ha considerado usar un String o un Integer, ambas opciones descartadas porque el atributo puede tomar exclusivamente los 6 valores indicados

Además, se ha permitido que este atributo sea nulo, para así poder considerar el caso en el que hayamos registrado un Code Audit y todavía no le hayamos relacionado ningún Code Record.

### **4.2.2 Entidad Code Record**

Se nos pide crear una entidad nueva llamada Code Record, la cual ya se ha mencionado previamente, pues es la entidad que compone a los Code Audit en el conglomerado.

El valor de los Code Audits se basa en el resultado de cada Code Record que lo compone; hará falta calcular la moda del valor del atributo mark de entre todos los Code Record que componen a un Code Audit para poder sacar el valor mark de este último.

De entre sus atributos cabe destacar la decisión de guardar el periodo como dos Date, fecha de inicio y fecha de fin, para poder luego calcular la diferencia entre ellos y así obtener el periodo en el que se ha realizado el Code Record.

### **4.2.3 Formulario Auditor Dashboard**

Se ha creado un formulario que contiene datos estadísticos sobre las entidades explicadas hasta el momento, Code Audit y Code Record. Al ser estos datos simples y rápidos de obtener con consultas SQL y operaciones muy poco complejas, se ha decidido que es más conveniente no persistir los atributos en la base de datos

### **4.2.4 Rol Auditor**

Para satisfacer un requisito suplementario, se ha desarrollado un nuevo rol: Auditor, el cual adoptarán los usuarios que realicen las inspecciones de código que hemos explicado previamente.

### **4.2.5 Datos de muestra**

Se han creado una amplia variedad de datos de muestra para cada entidad desarrollada, intentando cubrir rangos amplios y adecuados; siempre dentro de las restricciones e intentando cubrir tantos casos distintos como sea posible. Por ejemplo; en el campo mark de Audit Record se ha probado el mayor valor, el menor valor y valores intermedios, en campos de fechas se han probado fechas alejadas y cercanas al presente y en campos de texto se han probado distintos alfabetos.

Además, se han creado dos cuentas con rol de auditor: auditor1 y auditor2.

## 4.3 Entregable 3

Para satisfacer los requisitos de esta entrega, se han desarrollado una serie de repositorios, servicios, controllers en backend, y formularios y listas en frontend, que nos permiten llevar a cabo operaciones de listado, mostrar detalles, creación, edición, borrado y publicación.

A continuación, se va explicar resumidamente el proceso de desarrollo, haciendo hincapié en las decisiones de diseño que se han tomado, exponiendo para ello las alternativas consideradas junto con sus ventajas y desventajas, y finalmente el motivo de la elección.

### **4.3.1 Operaciones de *Auditors* sobre *CodeAudits***

Se han desarrollado, tanto a nivel de backend como de frontend, el código necesario para que los usuarios autenticados con el rol de *Auditor* puedan llevar a cabo las operaciones de listado, mostrado, creación, edición, publicación y borrado de las entidades *CodeAudits* o audiciones de código.

A la hora de hacer el listado, se ha decidido mostrar de cada *CodeAudit* las propiedades *code*, *execution date*, *type*¸ *project* y *draft mode*. En esta misma pantalla de listado se ha decidido ubicar el botón de creación.

Para mostrar una audicón, basta con clicar en una del listado. Esto nos lleva a un formulario donde se muestan los detalles de dicha audición en concreto, esto es, se muestran todas sus propiedades: *code*, *execution date*, *type*, *corrective actions*, *mark*, *link* y *project*. Aquí cabe destacar la decisión de mostrar el atributo type como un campo de texto “textbox”, en lugar de usar un desplegable, como quizá sea la decisión más obvia para un enumerado. Esto se debe a que dicha propiedad no se puede modificar directamente desde este formulario, sino que depende de los valores de los *AuditRecords*, como veremos más adelante.

Desde este mismo formulario podemos actualizar, borrar o publicar (cambiar la propiedad *draft mode* a true), y acceder a sus *Audit Records*. El botón de creación mencionado previamente también nos lleva a este formulario vacío. Aquí se tendrán en cuenta las restricción pertinentes.

Otra decisión de diseño sobre las operaciones de los *CodeAudits*, es la de que no sea posible publicarlos si alguno de sus *Audit Records* no está publicado, o si directamente no tiene ninguno.

La otra posibilidad considerada ante esta tesitura ha sido no aplicar ninguna restricción de este tipo, pero esto no resultaría coherente, puesto que hasta que no se finalicen todos los registros de audiciones, no damos por terminada una audición, además, la nota final viene dada por las notas individuales de cada registro, por lo que no tiene sentido adjudicar una nota definitiva hasta que no se terminen de poner las de los registros. Por todo esto, hemos descartado la mencionada posibilidad. Además, se ha llegado a esta conclusión con ayuda del foro de la asignatura, donde se ha abordado el problema.

### **4.3.2 Operaciones de Auditors sobre Audit Records**

Con los *Audit Records*, se ha procedido de forma similar, en el listado se enseña de cada uno su *code, mark* y *draft mode*, y es aquí donde se ha ubicado el botón de crear. En el mostrado de *Audit Records*¸ se pueden ver las propiedades *code*, *start date*, *end date*, *mark* y *link*. Aquí tenemos la posibilidad de borrarlo, actualizarlo y publicarlo, de nuevo, teniendo en cuenta las restricciones pertinentes.

Aquí, para mostrar en el formulario el atributo *mark* se ha optado por un desplegable en el que podemos elegir *A+*, *A*, *B*, *C*, *F*, *F-*, los cuales son las posibles notas que puede tener un registro. En este caso en el que sí que podemos introducir un valor, un desplegable resulta la mejor opción para un enumerado.

A nivel de decisiones, no hay nada más destacable en las operaciones de los Audit Records, aunque se ha encontrado una dificultad a nivel de implementación a la hora de recibir los datos de *mark*, pues esta, aunque al usuario se muestra como *A+* o *F-*, el enumerado admite *A\_PLUS* y *F\_MINUS* respectivamente, siendo los valores previos el displayValue. Para solucionarlo, simplemente se ha implementado una función para parsear de cualquier String a AuditMark, y este parseo se aplica en los binds pertinentes.

### **4.3.3 Auditor Dashboard**

Los auditores pueden acceder a una página donde se muestran una serie de estadísticas variadas relativas a sus *Code Audits* y *Audit Records* que están publicados, tales como el número de *Code Audits* con un determinado type, o el número promedio de *Audit Records* pertenecientes a sus *Code Audits*.

No ha habido decisiones de diseño a destacar

### **4.3.4 Registrarse y convertirse en Auditor**

Se ha añadido el botón para cualquier usuario registrado de convertirse en Auditor. Para ello, se debe rellenar un formulario con los datos de los auditores: *firm*, *professional* *ID*, *certifications* y *link*, de nuevo, teniendo en cuenta las restricciones pertinentes.

Adicionalmente, desde este mismo formulario, los auditores pueden actualizar la información de su perfil, cambiando los valores de los atributos mencionados previamente.

### **4.3.4 Operaciones de cualquier usuario sobre Code Audits**

Para cualquier usuario registrado, se ha añadido la posibilidad de acceder al listado de todos los *Code Audits*, mostrando de cada uno su *code*, *execution date*, y *type*. Como se puede ver, se ha optado por mostrar en este listado público menos información de cada entidad, porque se entiende que, al ser un listado de todas las audiciones que guarda el sistema, puede llegar a haber muchas entradas, y, de esta forma, se hace la información más escueta.

De forma análoga al resto de listados que se han expuesto, al clicar en una entidad del listado, se navega a un formulario donde se muestran los detalles de dicha entidad en concreto. Aquí se muestra el *code*, *execution date*, *type*, *corrective actions*, *mark*, *link*, *draft mode* y *project*. Naturalmente, al tratarse de una audición ajena, no se pueden modificar los datos.

# 5. Conclusión

Este documento está actualmente en su tercera versión, correspondiente al tercer entregable. Su desarrollo ha consistido principalmente en la creación de diferentes clases; dos entidades, un formulario y un rol, y en la creación de datos de muestra.

Este desarrollo ha dejado listo todo lo necesario para la implementación de las explicadas inspecciones de código a nivel de base de datos.

# 6. Bibliografía

Intencionadamente en blanco.