Analysis Report D02

INFORMACIÓN DE GRUPO

Group: C1.043

Repository: https://github.com/DaniFdezCab/DP2-2324-C1-043.git

Student #1
UVUS: danfercab
Contact: danfercab@alum.us.es

Student #2
UVUS: alvmarmun1
Contact: alvmarmun1@alum.us.es

Student #3
UVUS: pabberima
Contact: pabberima@alum.us.es

Student #4
UVUS: alepingar
Contact: alepingar@alum.us.es

Student #5
UVUS: fracapgar1
Contact: fracapgar1@alum.us.es

Date: Sevilla Febrero 08/03/2024

TABLA DE CONTENIDO

INTRODUCCIÓN

Este documento presenta un análisis de los requisitos para un sistema de gestión de código. El sistema tiene como objetivo centralizar el almacenamiento, la gestión y el seguimiento de las auditorías de código en un proyecto.

Se realiza la implementación de *code audits, audit records, auditor dashboard* y *auditor* como entidades en el proyecto. Con ello almacenamos, gestionamos y generamos informes sobre auditorías de código y sus registros para supervisar la calidad del código del proyecto

CONTENIDOS

Entidades y Atributos

El sistema gestiona dos entidades principales:

- **CodeAudit:** Este requisito de información se encargaría de almacenar todo la información referente a la auditoria del codigo la cual se ve reflejada en los siguientes atributos:
    - code (único, [A-Z]{1,3}-[0-9]{3})
    - executionDate (pasado)
    - type ("Estático", "Dinámico")
    - proposedCorrectiveActions (no vacío, < 101 caracteres)
    - mark (calculada como la moda de las calificaciones en los codeRecords)
    - optionalLink (información adicional)

Se analizaron los requisitos y se optó por modelarse de esta forma asegurando que se cumpliese con todas las restricciones implantadas por el cliente.

La tarea se realizó exitosamente excepto el cálculo para obtener mark ya que todavia no se puede implementar con los conocimientos actuales

- **CodeRecord:** Este requisito de información se encargaría de almacenar todo la información referente a los registros del código la cual se ve reflejada en los siguientes atributos:
    - code (único, AU-[0-9]{4}-[0-9]{3})
    - auditPeriodStart y auditPeriodEnd (pasado, mínimo 1 hora)
    - mark ("A+", "A", "B", "C", "F", "F-")
    - codeAudit (Many To One, relación con CodeAudit)
    - furtherInformation (información adicional,URL)

Se analizaron los requisitos y se optó por modelarse de esta forma asegurando que se cumpliese con todas las restricciones implantadas por el cliente.

La tarea se realizó exitosamente excepto el cálculo para obtener *haPasadoAlMenosUnaHora()* ya que todavia no se puede implementar con los conocimientos actuales.

También se crearon los clases tipo enumerado Mark con los valores (***A_PLUS***, ***A***, ***B***, ***C***, ***F***, ***F_MINOR***) y Type (***STATIC***, ***DYNAMIC***) para cumplir con los requisitos de las diferentes clasificaciones de marcas y de tipos de auditorías del código.

- **AuditorDashboard:** esta clase no está terminada ya que no se puede completar con los conocimientos actuales pero aquí están sus atributos:
    - `totalCodeAudits`
    - `averageAuditRecords`
    - `deviationAuditRecords`
    - `minimumAuditRecords`
    - `maximumAuditRecords`
    - `averagePeriodAuditRecords;`
    - `deviationPeriodAuditRecords;`
    - `minimumPeriodAuditRecords;`
    - `maximumPeriodAuditRecords;`

Todos son de tipo double excepto el primero, `totalCodeAudits` que es de tipo entero.

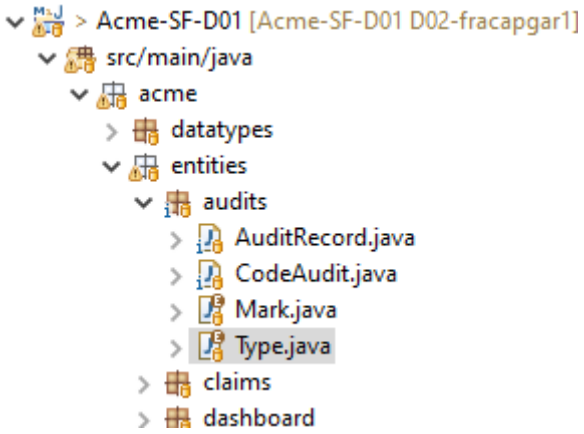
Roles nuevos añadidos al sistema:

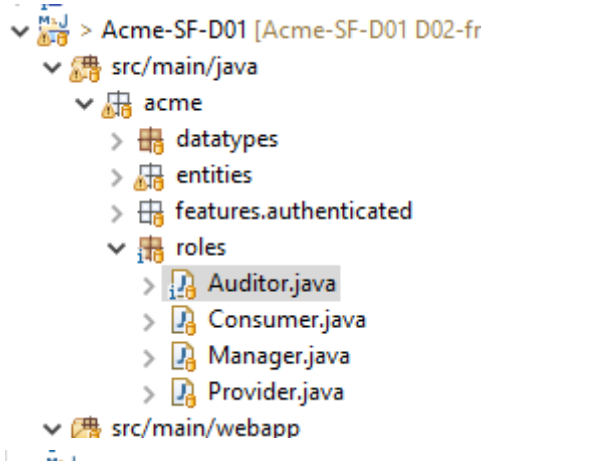
- **Auditor:**
    - firm (no vacío, < 76 caracteres)
    - professionalID (no vacío, < 26 caracteres)
    - certifications (no vacío, < 101 caracteres)
    - optionalLink (información adicional,URL)

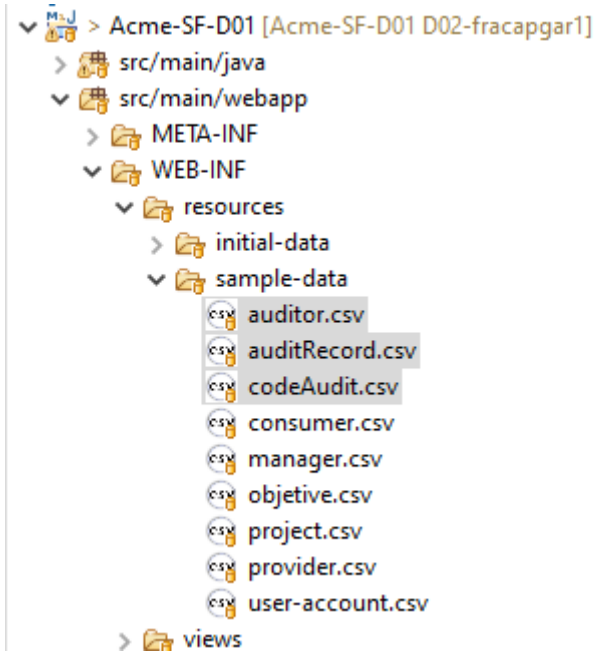
Para este nuevo rol se crearon dos nuevos perfiles auditor1 y auditor2 cumpliendo así con los requisitos de prueba.

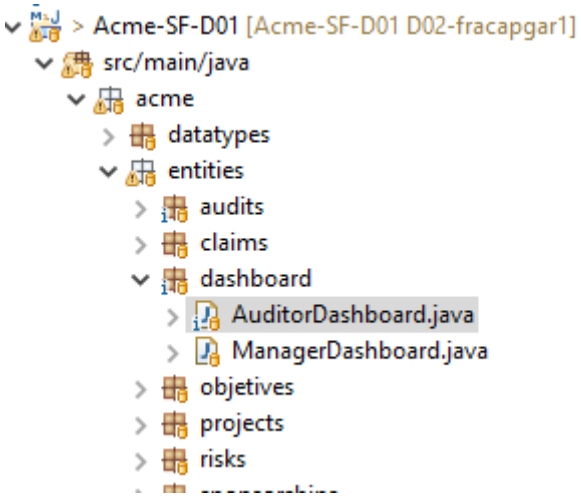
Además que para todas las entidades nuevas creadas se crearon sus respectivos csv con datos de prueba para comprobar si sus restricciones se encuentran bien implementadas y para encontrar posibles fallos en el código.

Este sería el explorer de lo mencionado anteriormente, visto desde eclipse:









CONCLUSIONES

Todas las clases fueron creadas sin problemas, pero hay apartados que no se han podido completar simplemente se han dejado señalado que con los conocimientos

actuales no se ha podido continuar con su implementación además que para asegurar un buen funcionamiento de las restricciones impuestas en los atributos sería recomendable ampliar la gama de ejemplos impuestos en los csv.