Universitat Oberta de Catalunya

uoc.edu

Diseño e implementación de la base de datos para una aplicación de control de procesos de seguridad informática.

Raúl Alonso Sihuro
Grado Ingeniería Informática
Bases de datos

Jordi Ferrer Duran
Xavier Baró Solé

Junio 2022

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © AÑO TU-NOMBRE.

Permission is granted to copy, distribute and/or modify this document under the terms of the

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño e implementación de la base de datos para una aplicación de control de procesos de seguridad informática.*
Nombre del autor:	*Raúl Alonso Sihuro*
Nombre del consultor/a:	*Jordi Ferrer Durán*
Nombre del PRA:	*Xavier Baró Solé*
Fecha de entrega (mm/aaaa):	06/2022
Titulación::	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Bases de Datos*
Idioma del trabajo:	Castellano
Palabras clave	*Oracle, vulnerabilidad informática, seguridad informática*

Resumen del trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El siguiente trabajo fin de grado tiene como objetivo la creación el diseño e implementación de una base de datos de control de procesos de seguridad informática. Descomponiendo la aplicación llegamos a la conclusión de que es necesario dar solución a una serie de necesidades planteadas que se dividen en cuatro bloques:

- Vulnerabilidades y mitigaciones.
- Empleados, departamentos y formaciones.
- Incumplimientos de políticas establecidas por la empresa.
- Sistema propuesto de log.

Una vez concluido el diseño se nos plantea otro gran desafío que es la explotación del modelo propuesto con un set de datos significativo, en el que podremos comprobar la validez del modelo, su adecuación a lo demandado y la escalabilidad del mismo.

Hemos utilizado scrum, a la hora de realizar el proyecto. Eso sí, adecuándose imaginativamente a este proyecto debido a su corta duración y escaso número de componentes del equipo participante (solo uno).

Dentro de los entregables que se incluyen junto a esta memoria, se encuentra:

- El diseño de la base de datos creada.
- Los procedimientos almacenados creados para el manejo de datos, así como los diferentes juegos de pruebas ejecutados.
- La respuesta a las consultas planteadas en el enunciado del TFG.

Finalmente hemos creado un sistema de log de transacciones, tal coo el enunciado nos propone, para tener controlada la ejecución de los diferentes procedimientos y la salida que han generado.

Concluimos creyendo que hemos dado respuesta a las necesidades planteadas y académicamente ha sido muy enriquecedor poner en práctica lo aprendido en el Grado.

Abstract (in English, 250 words or less):

The objective of the following final degree project is the creation, design and implementation of a computer security process control database. Breaking down the application we came to the conclusion that it is necessary to provide a solution to a series of needs that are divided into four blocks:

Vulnerabilities and mitigations.

Employees, departments and training.

Non-compliance of policies established by the company.

Proposed log system.

Once the design has been completed, another great challenge is the exploitation of the proposed model with a significant dataset, in which we can check the validity of the model, its adequacy to the requirements and its scalability.

We have used scrum, at the time of realizing the project. However, imaginatively adapting to this project due to its short duration and small number of participating team members (only one).

Among the deliverables included in this report, we find:

The design of the database created.

The stored procedures created for data management, as well as the different test sets executed.

The answer to the queries posed in the TFG statement.

Finally, we have created a transaction log system, as proposed in the statement, to control the execution of the different procedures and the output they have generated.

We conclude believing that we have responded to the needs raised and academically it has been very enriching to put into practice what we have learned in the Degree.

Agradecimientos:

Nunca he sido muy partidario de esta sección incluso en lo que no son libros de texto, cuando he visto expuesta la sección agradecimientos en presentaciones siempre me ha provocado cierto rechazo, no obstante tras seis años de esfuerzo continuo, veo obligatoria esta sección.

A Maria, mi compañera de vida, que es la persona que siempre me anima a tomar buenas decisiones, también inscribirse en la UOC fué una de ellas. Ella, que durante estos seis años se ha echado a sus espaldas más labores de las que le correspondían por dejarme a mi tiempo para estudiar. A ella, que ha aguantado mis desaires, cuando el agobio por los estudios, casa y trabajo, aparecían. GRACIAS, sin tí no hubiera sido posible.

También quiero dar las gracias a mis dos chavales, por su eterna paciencia conmigo y perdón por esas horas de ausencia en sus partidos los fines de semana.

Gracias a esta familia tan excepcional que tengo la suerte de tener, sin ellos llegar al final del Grado no hubiera sido posible.

Índice

1. Introducción

1.1 Contexto y justificación del Trabajo

En los últimos años se ha producido un proceso de digitalización en las empresas, nunca visto antes. En aras de aumentar la productividad se han automatizado gran cantidad de procesos de negocio haciendo estos más fáciles y eficientes.
Este proceso no tiene vuelta atrás y hace que aparezcan nuevos riesgos en seguridad, puesto que los nuevos requisitos y necesidades de comunicación implican mayor exposición de la información.

Se podría definir la ciberseguridad como una capa de protección para los archivos de información. Y, a partir de ella, se trabaja para evitar amenazas y riesgos para la información procesada, transportada y almacenada.

La ciberseguridad o seguridad digital tiene por tanto que jugar un papel fundamental en estas empresas donde todos sus sistemas se encuentran interconectados y con procesos de negocio que en gran medida necesitan exponer servicios en internet. Pese a esta importancia son pocas las empresas que tienen una política sobre ciberseguridad claramente definida, poniendo en un claro peligro su actividad empresarial e incluso la propia continuidad su negocio.

He aquí la justificación para la realización del tema sobre el que versa este TFG, el diseño e implementación de la base de datos para una aplicación de control de procesos de seguridad informática, con ella la compañía establecerá una línea de base para evaluar la cobertura de sus medidas de seguridad. Tomar conciencia de los peligros a los que estamos expuestos desglosando los activos tecnológicos pertenecientes a cada proceso de negocio para a partir de ahí, habiendo tomado conciencia de los asset peligrosos, empezar a trabajar para remediar ese posible peligro. Poniendo nombre y apellidos a la vulnerabilidad, donde la tenemos localizada, quién será el responsable de solucionarla y el grado de avance en su remediación. El objetivo de nuestra aplicación sería no solamente es aplicar diferentes sistemas de seguridad con el fin de prevenir y/o contrarrestar dichas vulnerabilidades, sino que también es educar a los usuarios sobre cómo evitar riesgos innecesarios.

1.2 Objetivos del Trabajo

El primer objetivo del proyecto de implantación de esta aplicación de control de procesos de seguridad informática será el análisis de vulnerabilidades. Se identificarán los distintos procesos de negocio de la empresa y se definirán los distintos activos tecnológicos que forman parte de dicho proceso. Una vez tengamos el glosario de vulnerabilidades y el departamento donde se encuentra asignaremos un responsable para el seguimiento de su remediación.

El segundo es llevar un registro exhaustivo de todas las sesiones de formación, tanto presenciales como telemáticas, que se realicen en la empresa referentes a temas de seguridad.

Y por último, la aplicación debe permitir gestionar cada una de las diferentes auditorías de seguridad definidas por la empresa.

El trabajo ha de ofrecer los diferentes resultados de sus consultas se definan en tiempo constante 1.

1.3 Enfoque y método seguido

Dado que el objetivo del TFG es la creación de una aplicación que cumpla con la funcionalidad básica del sistema analizado, se ha decidido aplicar una metodología adaptada. Durante el grado hemos trabajado con dos tipos de metodologías diferentes en varias asignaturas: las metodologías clásicas o pesadas, concretamente Waterfall y las metodologías ágiles, concretamente Scrum. Inicialmente pensamos seguir una metodología basada en metodologías pesadas, pero al llegar al punto de plantear el análisis se decide acercarnos más a las metodologías ágiles tan en boga en los tiempos que corren.

Scrum
Scrum es una metodología de trabajo para la gestión y desarrollo de software. Está basada en un proceso incremental e iterativo. Éste es un proceso básico diferenciándose de los clásicos. Se definen unos puntos de control para mejorar el proceso, lo que se conoce como retrospectivas. Está orientada hacia las necesidades del cliente, estando preparada para los cambios que puedan surgir durante el desarrollo del proyecto.

Esta metodología se caracteriza por entregar al cliente el software funcional en cada iteración, lo que permite comprobar la calidad del mismo e ir mejorando con un coste bajo aquellos aspectos que sea necesario corregir, ya que se encuentran las mejoras durante el desarrollo y no al final.

La documentación sería la mínima necesaria para permitir una buena gestión del proyecto, tal como indican los principios de scrum, pero ya que gran parte de la evaluación de este TFG se basa en la memoría, nos permitiremos la licencia de hacer una salvedad en este apartado. Esta adaptación también nos la permite realizar scrum, no existe nada de la metodología que no podamos adaptar al éxito del producto, en este caso un TFG aprobado.Con estas metodologías el cliente se compromete más con el proyecto, ya que lo ve crecer en cada iteración. Además, le permite hacer modificaciones funcionales o de prioridad de los requisitos del software al comienzo de cada iteración, pudiendo surgir estas modificaciones al ir viendo cómo se desarrolla y qué necesita más en él, con el fin de enfocarse mejor hacia los objetivos.

Generalmente, las metodologías ágiles son procesos iterativos en los que se entrelazan la especificación, el desarrollo, el diseño y las pruebas. El software se desarrolla incrementalmente, de tal manera que en cada incremento se incluyen nuevas funcionalidades al sistema.

En conclusión, comparando las metodologías ágiles con las pesadas: Un Software funcionando tiene mayor prioridad que una documentación exhaustiva y extensa. Los individuos y sus iteraciones son más importantes que los procesos y herramientas, la respuesta ante los cambios es más importante que el seguimiento de un plan y la colaboración con los clientes es más importante que la negociación de los contratos.

1.4 Planificación del Trabajo

Basándonos en dicha metodología ágil, se divide el tiempo que queda hasta la entrega del TFG en sprints de 14 días naturales haciendo coincidir los finales de sprint con los entregables a remitir a la Universidad.

Cada sprint se divide a su vez en cuatro fases: planificación del sprint, seguimiento del sprint, revisión del sprint y retrospectiva del sprint.

PLANIFICACIÓN DEL SPRINT
Es la reunión que se realiza al inicio de cada sprint, donde se define la lista de los requisitos del sistema. En cada iteración, esta lista es revisada y se seleccionan los objetivos, se identifica y comunica el trabajo que se realizará en esa iteración, y el tiempo que conllevará hacer dicho trabajo. El límite de duración es ocho horas.

SEGUIMIENTO DEL SPRINT
En esta fase tienen lugar los Daily Scrum o Stand-up meeting. Son las reuniones diarias en las que se va comprobando el avance de las tareas que se deben realizar durante el sprint. Estas reuniones se deben realizar siempre en el mismo lugar y a la misma hora, con una duración fija de 15 minutos, en las que cada miembro del equipo ha de contestar tres preguntas:

-¿Qué has hecho desde ayer?
-¿Qué es lo que harás hasta la reunión de mañana?
-¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?

REVISIÓN DEL SPRINT
Es la reunión que se tiene con el cliente al finalizar cada sprint con el fin de revisar el mismo. En ella se revisa el trabajo que ha sido completado y el que no ha podido serlo, presentando el trabajo completado a los interesados. Esta demo es importante ya que mejora el feedback con el cliente, habiendo un reconocimiento del trabajo avanzado, y se tiene la posibilidad de corregir los puntos más débiles para el siguiente sprint. El límite de duración son cuatro horas.

RETROSPECTIVA DEL SPRINT
Una vez superado cada sprint, se lleva a cabo una retrospectiva, es decir, todos los miembros del equipo comentan sus impresiones sobre el sprint recién acabado. El propósito de esto es la mejora continua del proceso. Su tiempo de duración son cuatro horas.

Adecuando a nuestro proyecto, en el que solo una persona forma parte del proyecto, se eliminará la reunión de seguimiento diaria ya que no tendré que informar a más personas cuál será el grado de avance del trabajo diario.
Dispondremos de flexibilidad a la hora del cierre de sprint a excepción, claro está, de las ceremonias de entrega de las PEC. Una vez leído el enunciado del TGF, determinamos que necesitaremos trabajar en él 3 días en semana a razón de 2 horas por día. Hemos dejado un margen de una hora por día que podemos utilizar en caso de necesidad.

1.4.1 Hitos y tareas.

Procederemos a dividir la planificación en dos partes, la primera será la planificación de cara a asegurar las entregas y la segunda será la división en tareas y subtareas necesarias para la creación y entrega del producto.

Planificación de entregas:

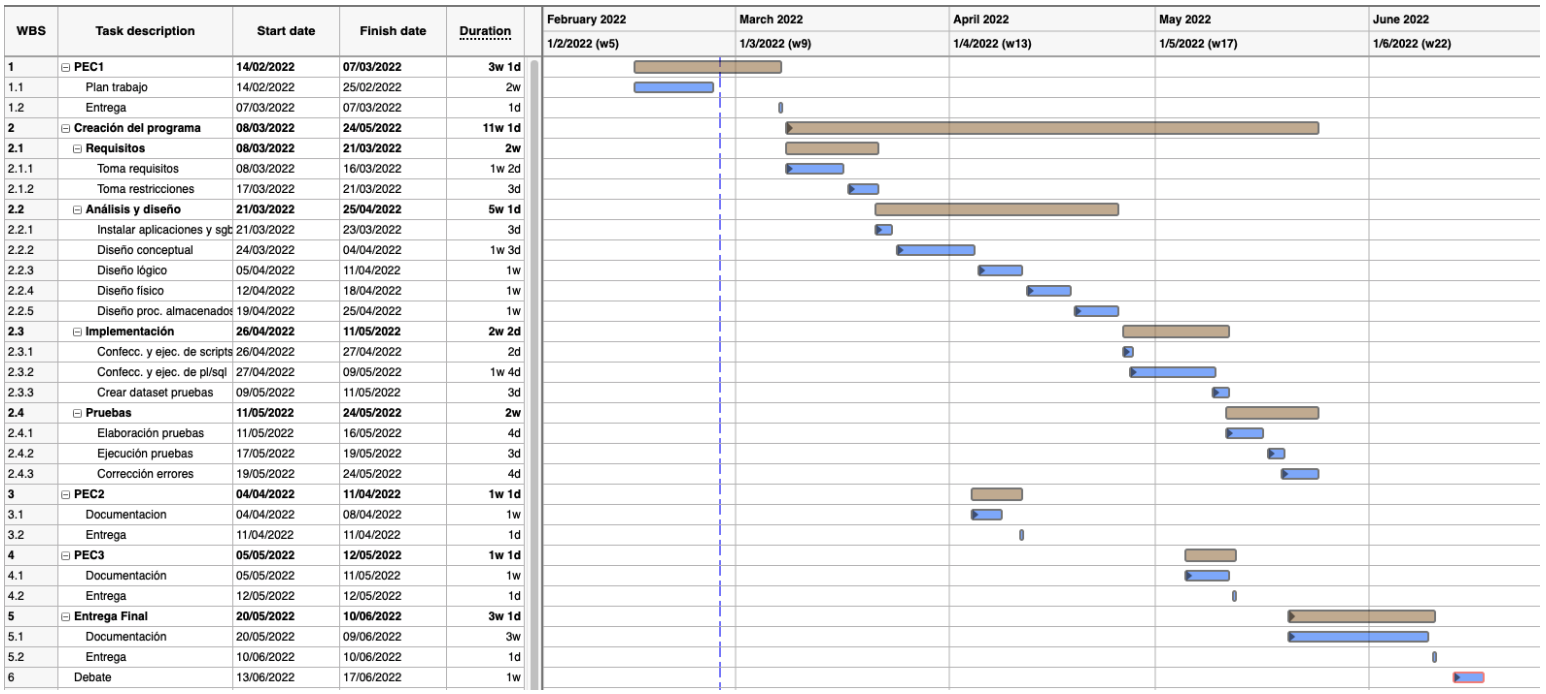
Nombre de la tarea	Duración	Comienzo	Fin
PEC1	**15 horas**	**21/02/2022**	**07/03/2022**
● Plan de trabajo	14 horas	21/02/2022	06/03/2022
● Entrega	1 horas	07/03/2022	07/03/2022
PEC2	**6 horas**	**04/04/2022**	**11/04/2022**
● Documentación	5 horas	04/04/2022	10/04/2022
● Entrega	1 hora	11/04/2022	11/04/2022
PEC3	**6 horas**	**05/05/2022**	**12/05/2022**
● Documentación	5 horas	05/05/2022	11/05/2022
● Entrega	1 hora	12/05/2022	12/05/2022
Entrega Final	**20 horas**	**20/05/2022**	**10/06/2022**
● Documentación	19 horas	20/05/2022	09/06/2022
● Entrega	1 hora	10/06/2022	10/06/2022
Debate Virtual	**5 días**	**13/06/2022**	**17/06/2022**

Planificación de creación y entrega del producto:

Nombre de la tarea	Duración	Comienzo	Fin
Definición de requisitos	**2 semanas**	**06/03/2022**	**07/03/2022**
● Toma requisitos.	1w 2d	08/03/2022	16/03/2022
● Toma restricciones.	3d	17/03/2022	21/03/2022
Análisis y diseño	**5w 3d**	**21/03/2022**	**25/04/2022**
● Instalar aplicaciones y sgbd	3d	21/03/2022	23/03/2022
● Diseño conceptual	1w 3d	24/03/2022	04/04/2022
● Diseño lógico	1w	05/04/2022	11/04/2022
● Diseño físico	1w	12/04/2022	18/04/2022

• Diseño procedimientos almacenados	1w	19/04/2022	25/04/2022
Implementación	**2w2d**	**26/04/2022**	**11/05/2022**
• Confección y ejecución scripts sgbd	2d	26/04/2022	27/04/2022
• Confección y creación pl/sql	1w4d	27/04/2022	09/05/2022
• Creación dataset pruebas	3d	09/05/2022	11/05/2022
Pruebas	**2w**	**11/05/2022**	**24/05/2022**
• Elaboración de pruebas	4d	11/05/2022	16/05/2022
• Ejecución de pruebas	3d	17/05/2022	19/05/2022
• Correcciones	4d	19/05/2022	24/05/2022

1.4.2 Diagrama de Gantt.



WBS	Task description	Start date	Finish date	Duration
1	PEC1	14/02/2022	07/03/2022	3w 1d
1.1	Plan trabajo	14/02/2022	25/02/2022	2w
1.2	Entrega	07/03/2022	07/03/2022	1d
2	Creación del programa	08/03/2022	24/05/2022	11w 1d
2.1	Requisitos	08/03/2022	21/03/2022	2w
2.1.1	Toma requisitos	08/03/2022	16/03/2022	1w 2d
2.1.2	Toma restricciones	17/03/2022	21/03/2022	3d
2.2	Análisis y diseño	21/03/2022	25/04/2022	5w 1d
2.2.1	Instalar aplicaciones y sgb	21/03/2022	23/03/2022	3d
2.2.2	Diseño conceptual	24/03/2022	04/04/2022	1w 3d
2.2.3	Diseño lógico	05/04/2022	11/04/2022	1w
2.2.4	Diseño físico	12/04/2022	18/04/2022	1w
2.2.5	Diseño proc. almacenados	19/04/2022	25/04/2022	1w
2.3	Implementación	26/04/2022	11/05/2022	2w 2d
2.3.1	Confecc. y ejec. de scripts	26/04/2022	27/04/2022	2d
2.3.2	Confecc. y ejec. de pl/sql	27/04/2022	09/05/2022	1w 4d
2.3.3	Crear dataset pruebas	09/05/2022	11/05/2022	3d
2.4	Pruebas	11/05/2022	24/05/2022	2w
2.4.1	Elaboración pruebas	11/05/2022	16/05/2022	4d
2.4.2	Ejecución pruebas	17/05/2022	19/05/2022	3d
2.4.3	Corrección errores	19/05/2022	24/05/2022	4d
3	PEC2	04/04/2022	11/04/2022	1w 1d
3.1	Documentacion	04/04/2022	08/04/2022	1w
3.2	Entrega	11/04/2022	11/04/2022	1d
4	PEC3	05/05/2022	12/05/2022	1w 1d
4.1	Documentación	05/05/2022	11/05/2022	1w
4.2	Entrega	12/05/2022	12/05/2022	1d
5	Entrega Final	20/05/2022	10/06/2022	3w 1d
5.1	Documentación	20/05/2022	09/06/2022	3w
5.2	Entrega	10/06/2022	10/06/2022	1d
6	Debate	13/06/2022	17/06/2022	1w

Hemos de destacar que esta planificación no utiliza las entregas de las PEC2 y PEC3 como hito en el que se ha de terminar una de las fases de la construcción del software. Tan solo va a ser un punto de control con el

consultor para comprobar el grado de avance y que no existan desviaciones importantes de la previsión inicial. A su vez estos puntos serán puntos donde recibiremos feedback sobre lo entregado hasta el momento, motivo por el cual hemos intentado que ya tengamos que entregar cierta cantidad de código en esa fecha.
La valoración de pesos por recursos no la hemos tenido que realizar al constar este proyecto con tan solo un recurso. Hemos tenido en cuenta en la duración de actividades la experiencia profesional de la que gozamos, instalar un Oracle no es una tarea compleja si se lleva años haciéndolo así como los detalles a nivel de infraestructura y uso del motor, el caso contrario lo encontraremos en las tareas de análisis que son en las que no poseemos tanta experiencia y que por tanto han sido beneficiadas con mayor número de días en el plan de trabajo.

1.4.4 Análisis de riesgos y planes de contingencia.

Hemos de tener en cuenta que en cualquier proyecto pueden surgir imprevistos que afecten a la planificación y realización del mismo. En proyectos cortos como el que hemos de realizar cualquier imprevisto por pequeño que parezca puede afectar al éxito y conclusión del mismo. Hemos de medir el impacto que estas situaciones inesperadas pueden tener en el desarrollo del proyecto y las medidas que utilizaremos para reducir el impacto sobre los entregables. Los riesgos más significativos identificados son:

Código	Nombre	Descripción	Impacto en planificación
01	Problemas de salud no controlables.	Enfermedad mía o de algún componente de mi familia.	ALTO
02	Inexactitud de las fechas planificadas.	Las fechas inicialmente planteadas en la planificación no se corresponden con la complejidad del proyecto, vista una vez comenzamos.	MEDIO
03	Problemas de aumento de carga laboral.	Aumento de la carga laboral que conlleva una reducción en horas disponibles para el proyecto.	MEDIO
04	Fallos en plataformas	Se imposibilita la continuación del	MEDIO

	hard/soft que se usan.	proyecto con modos óptimos por problemas de infraestructura o software.	
05	Pérdida de datos.	Problemas con los equipos utilizados o borrados accidentales hacen que perdamos contenido desarrollado.	MEDIO
06	No conectividad	Perdemos la conectividad con internet o el aula no está accesible.	BAJO
07	Retraso en las entregas de las PEC.	No conseguimos hacer las entregas establecidas en las diferentes PEC.	ALTO
08	Viaje de trabajo.	Existen viajes de trabajo que suelen ser de una duración de una semana fuera de España.	MEDIO

Para cada uno de los riesgos identificados hemos de proponer medidas mitigadoras que aseguren la correcta entrega del proyecto.
No podemos hacer que los riesgos no se conviertan en realidad, lo que sí que podemos hacer es establecer de una forma proactiva una serie de medidas para que cuando aparezcan estos problemas podamos mitigarlos inmediatamente y en caso de no poder hacerlos desaparecer, establecer los planes de contingencia que nos permitan minimizar su impacto.

Código	**Acción**	**Tipo**	**Riesgo después de aplicar la acción.**
01R1	Aumentar las horas de dedicación al proyecto después de la enfermedad, reservar días de vacaciones y hacer uso de ellos en este caso.	Mitigadora	Bajo
01R2	Pedir al consultor una ampliación	Correctora	Medio

	en las entregas de las PEC.		
02R1	Corrección de fechas en la planificación e incluso aumentando las horas de dedicación.	Correctora	Medio
03R1	Corrección de fechas en la planificación e incluso aumentando las horas de dedicación.	Correctora	Medio
04R1	Se dispone de otra máquina con Oracle XE y sqldeveloper instalado.	Correctora	Bajo
05R1	Para la generación de documentación y productos usamos la herramienta google drive que nos sincroniza al momento en el cloud los ficheros que generamos.	Correctora	Bajo
06R1	Se dispone de conectividad a internet por móvil, no da la misma calidad de conexión pero posibilita, de alguna manera, poder seguir trabajando.	Mitigadora	Bajo
07R1	Corrección de fechas en la planificación e incluso aumentando las horas de dedicación.	Correctora	Medio
07R2	Pedir al consultor una ampliación en las entregas de las PEC.	Correctora	Medio
08R1	Corrección de fechas en la planificación e incluso aumentando las horas de dedicación.	Correctora	Medio
08R2	Pedir al consultor una ampliación en las entregas de las PEC.	Correctora	Medio

1.5 Breve sumario de productos obtenidos

Cuando finalicemos el proyecto tendremos que hacer entrega de la siguiente serie de documentos:

- Memoria: Es la parte principal de la documentación a entregar, en ella haremos una descripción exhaustiva del desarrollo del proyecto e incluiremos también el diseño de base de datos realizado.
- Presentación: Presentación del proyecto donde se presenta la solución desarrollada.
- Autoinforme de evaluación: Registro de la evaluación de las competencias transversales.
- Producto: Incluirá los paquetes desarrollados en plsql de alta-baja-modificación, las pruebas de esos paquetes,las cargas de datos que haremos de prueba y los scripts de creación de la base de datos. Así mismo, incluiremos las queries que dan solución a las consultas que se nos pidieron desarrollar en el enunciado. Al estar fuera del alcance del proyecto el desarrollo de pantallas de la aplicación, hemos utilizaremos sql dinámico, sin el uso de funciones de agrupación, para proporcionar las distintas respuestas.

1.6 Breve descripción de los otros capítulos de la memoria

El primero de los capítulos será el de análisis de requisitos en el que trataremos los requerimientos funcionales y no funcionales del proyecto, estos serán la base del diseño de base de datos que hemos de realizar.

A continuación, tendremos que identificar las entidades, relaciones y atributos que formarán nuestra base de datos. Cuando hayamos realizado este análisis tendremos nuestro diseño conceptual.

En el capítulo del diseño lógico describiremos los datos con el mayor detalle posible, independientemente de cómo se implementarán físicamente en la base de datos. Normalizaremos y procederemos a asignar las claves primarias.

Finalmente llegaremos al capítulo del diseño físico donde nos ocuparemos de optimizar el rendimiento a la vez que aseguramos la integridad de los datos al evitar repeticiones innecesarias de los mismos. Durante el diseño físico, se transforman las entidades en tablas, las instancias en filas y los atributos en columnas.

Llegaremos a la parte de codificación del proyecto donde deberemos plasmar en código binario lo que sobre el papel hemos definido, una vez creadas ya las tablas en la base de datos. Empezaremos con la creación de los procedimientos almacenados de alta-baja-modificación que usaremos para el manejo de los datos que vamos a almacenar en el gestor. En esta fase crearíamos de ser necesarias las distintas funciones que nos ayuden en el manejo de comprobaciones, etc.

Seguidamente procederemos a definir y ejecutar el plan de pruebas, creando un conjunto de datos de datos de prueba que nos permita comprobar la oportuna funcionalidad de los paquetes creados. Una vez cargado este set de datos y después de probar los distintos procesos de ABM, pasaremos a dar respuesta a las diferentes consultas que se nos plantean en el enunciado, así como unas cuantas que añadiremos para dar solución a preguntas que se nos han ido ocurriendo mientras desarrollamos y que consideramos importantes como usuarios de la futura aplicación.

Y finalmente incluiremos las conclusiones, glosario y bibliografía utilizada durante el desarrollo del proyecto. Al final de la documentación será incluido un anexo con el proceso de instalación de Oracle XE y de Oracle Sql Developer, nuestro motor de base de datos y nuestra herramienta de desarrollo, respectivamente.

2. Resto de capítulos

2.1. Toma de requisitos

Llegamos a la fase del proyecto en el que tenemos que leer bien el enunciado para de él extraer las necesidades del proyecto. Es curioso ya que aquí nos encontramos el primer problema con la metodología que he elegido seguir que es scrum. ¿Cómo propone Scrum que debemos hacer esta “toma de requisitos"? Es curioso, como Scrum tiene muy poca propuesta en este aspecto, Scrum prefiere centrarse en la entrega, ir lo más rápido posible al mercado y recibir feedback. La fecha de entrega del producto total no es lo importante, por eso no nos dice cómo debe ser la toma de requisitos. Hemos de adaptar dicha metodología al plan de proyecto que nos hemos establecido, por ello y como lo que vamos buscando es la entrega de algo lo más rápido posible que dé valor al proyecto, algo que el usuario pueda ya empezar a probar, hemos decidido hacer lo siguiente.

En las fases de análisis y diseño, en lugar de hacer las entregas de todo en completo iremos haciendo las entradas de cada uno de los módulos en los que creemos podemos dividir la aplicación que se nos propone a saber:

- Vulnerabilidades y mitigaciones.
- Empleados y formaciones.
- Incumplimientos de políticas establecidas por la empresa.
- Sistema propuesto de log para hacer seguimiento a los posibles errores de introducción de datos de la aplicación.

No pensemos que al modularizar la aplicación no vamos a tener una visión completa del proyecto, siempre vamos a tener en mente el conjunto global de la aplicación, pero eso sí lo que vamos a intentar es ir proporcionando lo antes posible pequeñas funcionalidades de la aplicación en el menor tiempo posible,
Puede ser un poco más complicado a la hora de hacer el seguimiento de proyecto en las PEC intermedias ya que hasta casi el momento final de la última entrega no dispondremos de la visión total de todas las fases, pero a cambio conseguiremos tener componentes ya codificados antes de los que tendríamos con waterfall y conseguiremos el feedback del profesorado antes de que cometamos errores graves que nos hagan tener que decodificar gran parte de la aplicación.

2.1.1 Requisitos funcionales

En este apartado vamos a enumerar cuales son las necesidades planteadas por el proyecto y los datos que tenemos que almacenar en la base de datos para dar solución a estas demandas.

Código	Requisito funcional
REQ1	Almacenaremos las vulnerabilidades cada uno de los procesos de gestión de la empresa con el estado en que se encuentra y su importancia.
REQ2	Estableceremos y grabaremos la distinta relación entre el personal y departamentos.
REQ3	Por cada una de las vulnerabilidades, registraremos las acciones de mitigación necesarias para solventarla, un responsable, el estado en que se encuentra y su fecha tope de resolución.
REQ4	Registrar las políticas de seguridad de la empresa.
REQ5	Grabaremos cada uno de los incumplimientos de las políticas de seguridad, indicando el número de incumplimientos por departamento. A su vez, registraremos qué usuarios han incumplido alguna política con la fecha en la que lo hizo, sin guardar registro de lo que incumplió, para salvaguardar la privacidad.
REQ6	Una vulnerabilidad estará completamente mitigada cuando todas las acciones de mitigación se encuentren acabadas.
REQ7	Registraremos las distintas sesiones de seguridad, indicando su tipología y los empleados que han cursado las mismas.
REQ8	Se realizarán distintas auditorías de seguridad, internas o externas, en las que mediremos el cumplimiento de las diferentes políticas de empresa.
REQ9	Dar respuesta a las siguientes consultas: Departamento que, en un año concreto, tiene un número mayor de incumplimientos de seguridad registrados en la BD. Proceso de gestión interno que, teniendo en cuenta toda la información de que se dispone en la BD, ha tenido un mayor número de vulnerabilidades detectadas.

	Top5 de usuarios por número de incumplimientos asociados directamente a ellos, o a su departamento, durante el año en curso. Porcentaje de vulnerabilidades que, en el momento de ejecutar la consulta, están totalmente mitigadas. Número total de acciones de mitigación que, en el momento de ejecutar la consulta, no están totalmente acabadas. Política de seguridad que, en el momento de ejecutar la consulta, ha tenido más incumplimientos (teniendo en cuenta todos los departamentos de la empresa) Dado un determinado departamento de la empresa, y teniendo en cuenta el momento de ejecutar la consulta, porcentaje de usuarios del departamento que no han acabado todas las formaciones de seguridad asignadas. Porcentaje de usuarios de la empresa que, en el año en curso, no tienen ningún incumplimiento asignado. Teniendo en cuenta todas las auditorías externas realizadas, año en el cual se han detectado más incumplimientos (teniendo en cuenta sólo los detectados durante la auditoría). Porcentaje de vulnerabilidades críticas que, en el momento de ejecutar la consulta, tienen alguna acción de mitigación abierta (que no esté en estado “acabada”). Teniendo en cuenta el último año (el anterior al año en curso), título de la sesión formativa telemática que ha tenido un porcentaje menor de participantes en total. Número de vulnerabilidades críticas detectadas internamente teniendo en cuenta todos los datos de que se dispone. Se consideran detectadas internamente si se detectaron en posterioridad al análisis realizado al inicio del proyecto por la consultora externa. En el momento de ejecutar la consulta, porcentaje de acciones de mitigación en el sistema que están en los estados “en proceso” o “en revisión”. Teniendo en cuenta todas las acciones de mitigación en estado “en proceso”, persona responsable con más acciones asignadas.

2.1.2 Requisitos no funcionales

Código	Requisito no funcional
REQ1	La base de datos ha de ser relacional relacional.
REQ2	Usaremos Oracle.
REQ3	El manejo de la información se ha de hacer mediante procedimientos plsql de alta-baja-modificación.
REQ4	Las consultas que se nos plantean y que hemos expuesto en los requisitos funcionales han de ser respondidas usando tiempo constante. Es decir, mediante el uso de sql sin funciones de agrupamiento, vistas sencillas, vistas materializadas, etc.
REQ5	Tendremos que validar los datos que introducimos para que cumplan con lo especificado en el enunciado y la solución propuesta se ha de testear usando diferentes juegos de datos.
REQ6	Ha de ser una bbdd escalable y se ha de desnormalizar para que futuras modificaciones sean fáciles de realizar.
REQ7	El sistema almacenará información detallada de las llamadas a procedimientos de la base de datos mediante logs. Deberán disponer de un parámetro de salida RSP de tipo String que indicará si se ha finalizado correctamente (‘OK’) o si ha fracasado (‘ERROR+TIPO ERROR’) Se dispondrá de tratamiento de excepciones.

2.2 Diseño conceptual

El diseño conceptual de la base de datos no hay trabajo directo en un modelo de base de datos. El proceso es únicamente un ejercicio de identificación de datos relevantes.

Dos cosas principales que se identifican en el diseño conceptual de la base de datos son las entidades y las relaciones: las entidades son objetos reales en el mundo material y las relaciones son la red de conexiones que vinculan una entidad a otra indefinidamente.

2.2.1 Entidades y atributos

Vamos a definir las diferentes entidades y atributos que ha de contener el modelo conceptual. Tal y como hemos comentado vamos a ir haciendo estos estudios poco a poco, no pretendemos tener cerrado el modelo conceptual sino que iremos cerrando el proyecto por bloques para poder ir entregando productos funcionales lo antes posible a los usuarios. Recordemos que los bloques en los que hemos dividido el proyecto son:

- Vulnerabilidades y mitigaciones.
- Empleados, departamentos y formaciones.
- Incumplimientos de políticas establecidas por la empresa.
- Sistema propuesto de log para hacer seguimiento a los posibles errores de introducción de datos de la aplicación.

Empezaremos con el **bloque de vulnerabilidades y mitigaciones**:

Entidad	Descripción	Atributos
fabricantes	En esta entidad almacenaremos los detalles de los fabricantes con los que trabajamos en la compañía y que nos proporcionan los elementos software y hardware que usa en su realización de procesos.	↦ Código del fabricante. ↦ Nombre del fabricante. ↦ La denominación social del fabricante. ↦ Datos de contacto del fabricante.
procesos	Almacenamos los distintos procesos informáticos que la empresa necesita para la realización de sus operaciones.	↦ Código del proceso. ↦ Nombre del proceso. ↦ Descripción detallada del proceso.
maestrovulnerabilidades	Registramos las vulnerabilidades detectadas con los detalles de las mismas.	↦ Código de la vulnerabilidad. ↦ Nombre de la vulnerabilidad. ↦ Código del fabricante cuyo producto sufre la vulnerabilidad. ↦ Código del proceso que sufre la vulnerabilidad. ↦ Estado de la vulnerabilidad. ↦ Código CVEcode. ↦ Descripción de la vulnerabilidad. ↦ Referencias. ↦ En qué fecha se detectó. ↦ Código de importancia.
mitigaciones	Cada vulnerabilidad llevará asociadas acciones de mitigación que se crean para	↦ Código de la mitigación. ↦ Código de la

	arreglar la vulnerabilidad. Aquí mantendremos el registro de las acciones de mitigación creadas para las vulnerabilidades informadas.	vulnerabilidad. ↦ Pequeña descripción de la mitigación. ↦ Descripción de la mitigación. ↦ Código del empleado responsable de solventarla. ↦ Fecha límite en la que la acción de mitigación ha de concluir. ↦ Código del estado en el que se encuentra la mitigación.

Adicionalmente a las ya detalladas necesitaremos otras entidades de apoyo en este bloque que nos permitirán definir mejor las entidades principales:

Entidad	Descripción	Atributos
estadosvulne	Almacenaremos los distintos estados posibles de una vulnerabilidad. Lo que en nuestro argot es conocido como una tabla de lookup.	↦ Código de estado. ↦ Descripción del estado.
importancia	Registramos la importancia del impacto de las vulnerabilidades detectadas. Sería otra tabla de lookup.	↦ Código de la importancia. ↦ Descripción de la importancia.
estadomitigaciones	Registramos los distintos estados posibles de las mitigaciones. Sería otra tabla de lookup.	↦ Código de estado. ↦ Descripción del estado.

Continuaremos con el **empleados, departamentos y formaciones**:

Entidad	Descripción	Atributos
empleado	En esta entidad almacenaremos los datos de los empleados que trabajan para la compañía.	↦ Código del empleado. ↦ Código del departamento. ↦ NIF del empleado. ↦ Nombre del empleado. ↦ Apellido1 del empleado. ↦ Apellido2 del empleado. ↦ email del empleado. ↦ Fecha en la que fué alta el empleado en la compañía. ↦ Fecha en la que fué baja el empleado en la

		compañía.
incumplimientosempleados	Almacenamos si un empleado en cuestión ha tenido algún incumplimiento de las políticas de la compañía. No almacenamos el incumplimiento por cuestiones de privacidad.	↦ Código del incumplimiento. ↦ Código del empleado. ↦ Fecha en el que incumplió.
sesionesformativas	Registramos las sesiones a las que ha asistido cada uno de los empleados con la nota que han obtenido.	↦ Código del empleado. ↦ Código de la sesión formativa. ↦ Código la nota.
maestrosesiones formativas	Características de cada una de las sesiones formativas que hemos realizado o programado.	↦ Código de la sesión formativa. ↦ Título de la sesión. ↦ Agenda de lo que se va a dar en la sesión. ↦ Código del tipo de formación. ↦ Fecha inicio formación. ↦ Fecha fin formación.
departamento	Registraremos los distintos departamentos de la empresa.	↦ Código departamento. ↦ Nombre del departamento.

Adicionalmente a las ya detalladas necesitaremos otras entidades de apoyo en este bloque que nos permitirán definir mejor las entidades principales:

Entidad	Descripción	Atributos
notas	tipos de notas.Sería otra tabla de lookup.	↦ Código notas formación. ↦ Descripción notas de formación.
tipoformacion	tipos de formaciones que se han impartido.Sería otra tabla de lookup.	↦ Código tipo formación. ↦ Descripción del tipo de formación.

Procederemos a estudiar el bloque de **incumplimientos de políticas establecidas por la empresa y auditorías,** buscando las distintas entidades y atributos:

Entidad	Descripción	Atributos
incumplimientospoliticas	En esta entidad almacenaremos los distintos incumplimientos de las políticas por parte de todas las unidades de la empresa y que han sido aprobadas por rrhh.	↦ Código del incumplimiento. ↦ Código de la política. ↦ Código del departamento que ha incumplido política. ↦ Número de incumplimientos de esa política en ese departamento. ↦ Comentarios cortos a añadir si se estima. ↦ fecha de registro. ↦ comentarios.
maestropoliticasseguridad	Registramos todas las políticas de obligado cumplimiento que han sido aprobadas por rrhh y la dirección.	↦ Código de la política. ↦ Descripción corta. ↦ Descripción.
resultadosautoria	Copia de registro en un momento en el tiempo de los incumplimientosdepoliticas, haciendo una foto estática.	↦ Código incumplimiento. ↦ Código auditoría. ↦ Código de la política que se incumple. ↦ Código del departamento que incumple. ↦ Cuántas veces ha incumplido ese dpto esa política. ↦ Descripción corta. ↦ Fecha de registro. ↦ Comentarios. .
maestroauditoria	Registramos toda la información relativa a las auditorías.	↦ Código de la auditoría. ↦ Código del tipo de auditoría. ↦ Datos de quién ha realizado la auditoría. ↦ Fecha de la realización de auditoría.
tipoauditoria	Tipo lookup, indicamos qué tipo de auditoría se realiza.	↦ Código de tipo de auditoría. ↦ Descripción del tipo de auditoría.

Adicionalmente a las ya detalladas necesitaremos otras entidades de apoyo en este bloque que nos permitirán definir mejor las entidades principales:

2.2.2 Relaciones

A continuación y como hemos explicado anteriormente, empezaremos a identificar las relaciones entre las distintas entidades del **bloque de vulnerabilidades y mitigaciones**:

Entidades	Tipo de relación	Descripción
Procesos-maestrovulnerabilidades	1:0..N	Un proceso de negocio puede tener 0 o N vulnerabilidades. Una vulnerabilidad puede estar presente en varios procesos.
maestrovulnerabilidades-mitigaciones	1:0..N	Una vulnerabilidad puede tener 0 o N mitigaciones definidas. Una mitigación solo puede corresponder a una vulnerabilidad.
maestrovulnerabilidades-fabricantes	1:1	Una vulnerabilidad sólo puede corresponder a un único fabricante. Un fabricante puede tener varias vulnerabilidades.
mitigaciones-estadosmitigaciones	1:1	Una mitigación sólo puede tener un estado. Pero ese estado puede estar en varias mitigaciones.
maestrovulnerabilidades-estadosvulne	1:1	Una vulnerabilidad sólo puede tener un estado. Pero ese estado puede estar en varias vulnerabilidades.
maestrovulnerabilidades-importancia	1:1	Una vulnerabilidad sólo puede tener una importancia. Pero esa importancia puede estar en muchas vulnerabilidades.

Continuaremos identificando las relaciones entre las distintas entidades del **empleados, departamentos y formaciones**:

Entidades	Tipo de relación	Descripción
mitigaciones- empleado	1:1	Es el enlace con el bloque anterior. Una mitigación puede tener exclusivamente un responsable asignado, mientras que un responsable puede tener varias mitigaciones asignadas.

empleado- incumplimientoempleados	1:1	Sólo nos interesa saber si un usuario tiene incumplimientos o no.
empleado- sesiones formativas	1:0..N	Un empleado puede estar o no inscrito a alguna/s sesiones formativas. Una sesión formativa ha de tener al menos un usuario e ir hasta N usuarios.
sesionesformativas- notas	1:1	Una sesión formativa tiene que tener una nota, aunque sea simplemente inscrito. Una nota puede corresponder a muchos.
sesionesformativas- maestrosesionesformativas	N:N	Una sesión formativa que recibe un empleado puede corresponder a varios cursos. Mientras que al revés en una sesión maestro formativa puede haber muchos empleados inscritos..
maestrosesionesformativas- tipoformacion	1:1	Una formación puede ser externa o interna, pero no las dos a la vez. Mientras que puede haber muchas sesiones externas o internas.
empleado- departamento	1:1	Un empleado puede pertenecer sólo a un departamento. Un departamento puede tener muchos empleados.

Procederemos a estudiar las relaciones existentes en el bloque de **incumplimientos de políticas establecidas por la empresa y auditorías**:

Entidades	Tipo de relación	Descripción
maestropoliticasseguridad- incumplimientospoliticas	1:N	Una política tiene N incumplimientos asociados a las políticas por departamento. A su vez, un incumplimiento puede verse incumplido en N políticas.
incumplimientospoliticas- resultados auditorias	1:1	Es una copia registro a registro de la una a la otra. La misma relación existiría en sentido contrario.
marestroauditoria- resultadosauditoría	1:N	Una auditoría tiene N resultados. Al contrario un resultado pertenece a una única auditoría.
auditoria- tipoauditoría	1:1	Una auditoría tiene que ser sólo de un tipo. Al revés un tipo puede tener N auditorías asociadas..

2.2.4 Restricciones de integridad

Con carácter general a todo el diseño de base de datos cuando un atributo se llame idxxxxx han de ir siempre informados, no pudiendo ser nulos, por lo que no haremos más reseña sobre los mismos. Los atributos con nombre “desc” han de estar siempre informados y han de ser únicos.
Llegados a este punto hemos de averiguar las diferentes restricciones de integridad, que para el **bloque de vulnerabilidades y mitigaciones** son:

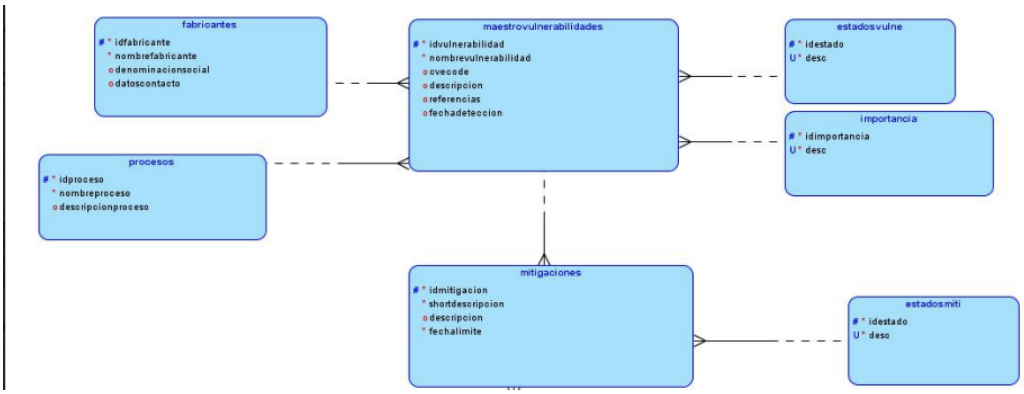
- El nombre del fabricante no puede ser nulo.
- El nombre de la vulnerabilidad no puede ser nulo.
- El nombre del proceso no puede ser nulo.
- La descripción corta de la mitigación no puede ser nula.
- La fecha límite que nos marcamos para mitigar la vulnerabilidad no puede ser nula y ha de ser superior a la del día de hoy.

Al igual que hemos comentado antes, cuando un atributo se llame idxxxxx han de ir siempre informados, no pudiendo ser nulos, por lo que no haremos más reseña sobre los mismos. Para el bloque de **empleados, departamentos y formaciones:**

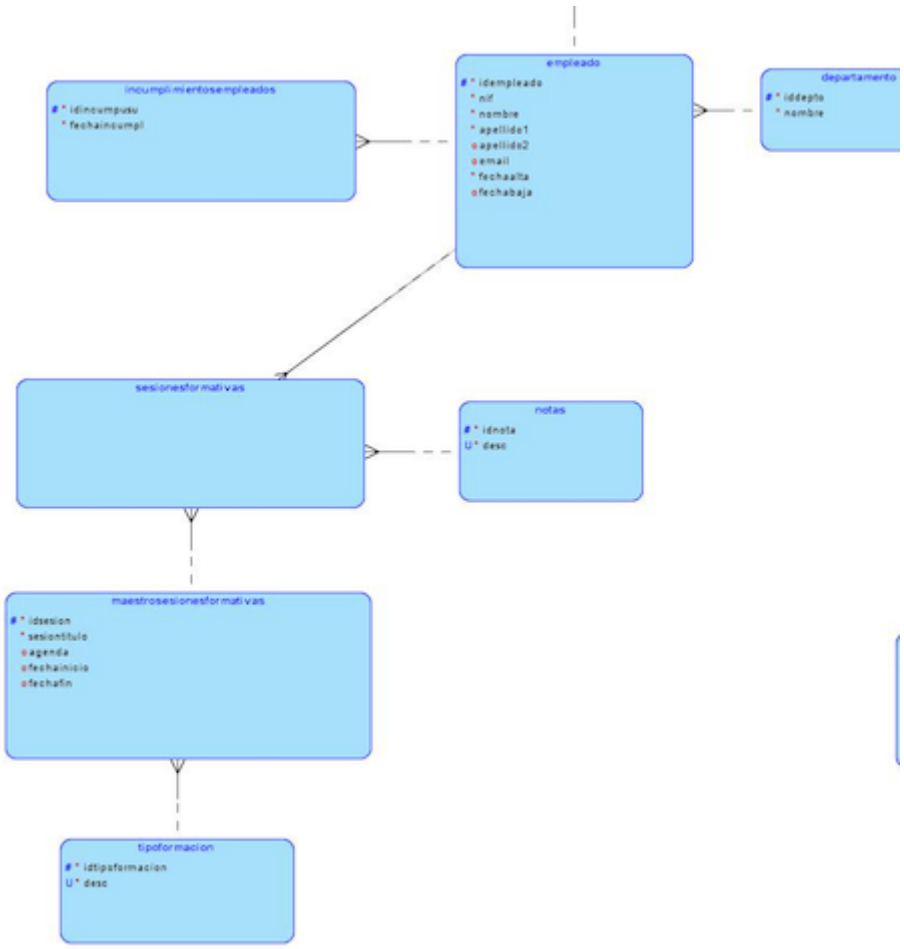
- El NIF del empleado no puede ser nulo y ha de cumplir que la letra dada corresponda a la calculada con la numeración proporcionada, pasándola por el algoritmo contenido en la función que proporcionamos en el producto.
- El nombre del empleado no puede ser nulo.
- El primer apellido no puede ser nulo, el segundo sí ya que muchos extranjeros no disponen de él.
- La fecha de alta del empleado en la compañía no puede ser nula.
- Ningún dato del incumplimiento del empleado puede ser nulo y la fecha del incumplimiento ha de ser menor que la actual.
- El nombre del departamento no puede ser nulo.
- El título de la sesión formativa no puede ser nulo.

2.2.4 Diagrama conceptual

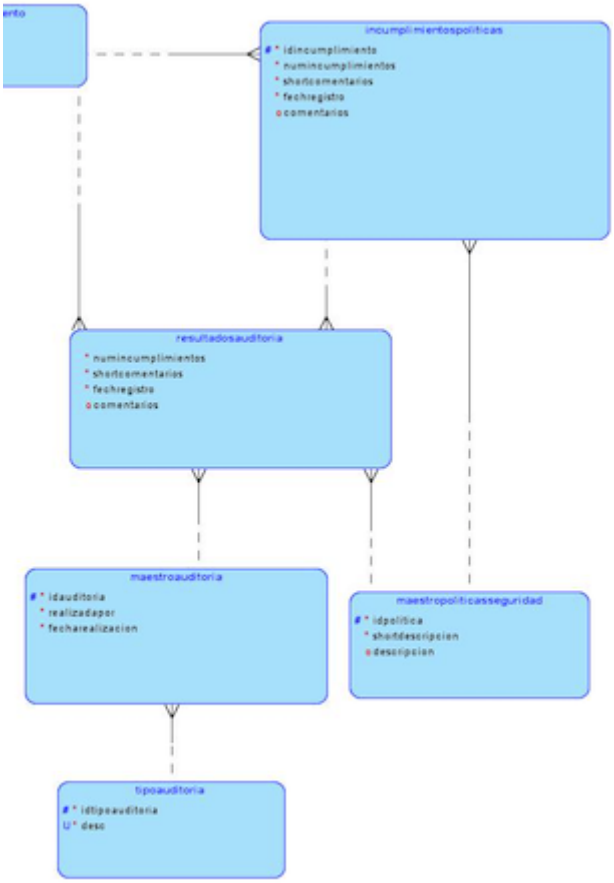
Para el **bloque de vulnerabilidades y mitigaciones** el diagrama conceptual sería:



Para el bloque de **empleados, departamentos y formaciones** el diagrama conceptual sería:

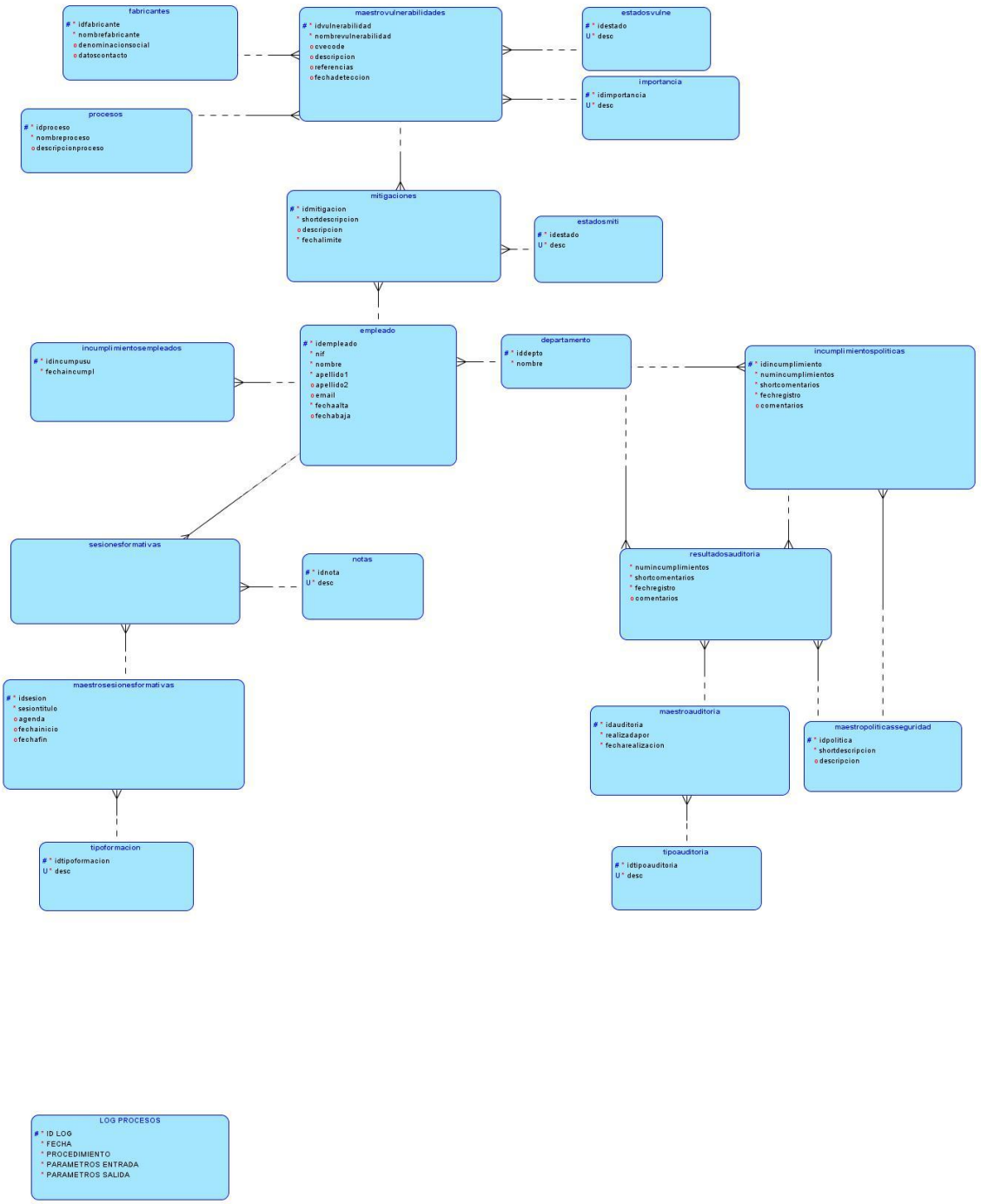


Para el bloque de **incumplimientos de políticas establecidas por la empresa,** este sería su modelo conceptual:



Para el bloque de **log,** este sería su modelo conceptual:

Agrupando todos los bloques, este sería su modelo conceptual:



2.3 Diseño lógico

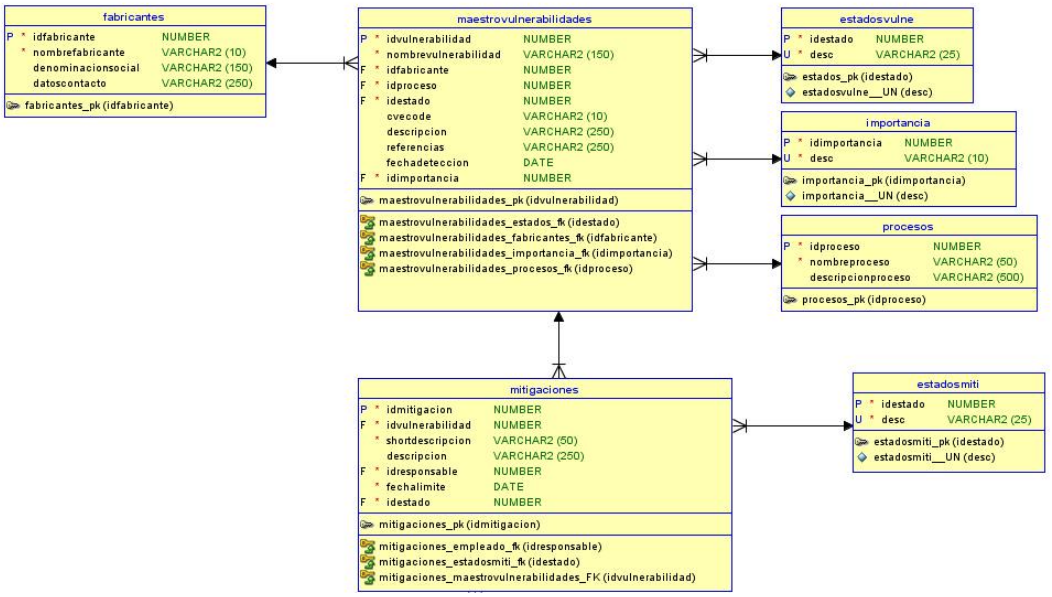
El objetivo del diseño lógico es convertir los esquemas conceptuales locales en un esquema lógico global que se ajuste al modelo de base de datos sobre el que se vaya a implementar el sistema. Mientras que el objetivo fundamental del diseño conceptual es la compleción y expresividad de los esquemas conceptuales locales, el objetivo del diseño lógico es obtener una representación que use, del modo más eficiente posible, los recursos que el

modelo de base de datos posee para estructurar los datos y para modelar las restricciones.

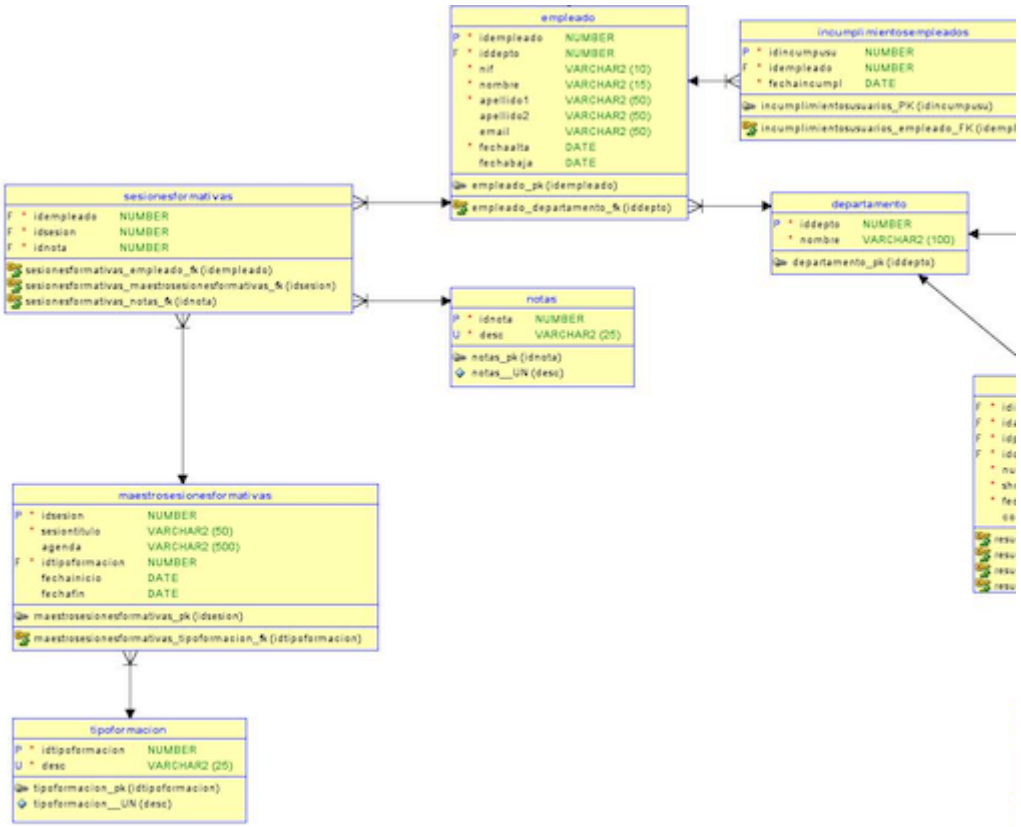
Lo primero que hay que realizar en la fase de diseño lógico, es obtener el esquema lógico estándar, a partir del esquema conceptual obtenido en la primera fase. Las reglas que permiten pasar del modelo E/R al esquema lógico, son las que a continuación se explican:

Cada entidad se transforma en una tabla y los atributos de dicha entidad en atributos de la tabla. Esto es, cada entidad genera una tabla, con sus mismos atributos, incluyendo las claves.
Las relaciones de muchos a muchos se transforman en tablas cuya clave estará formada por la clave primaria de las entidades relacionadas, convirtiéndose ésta en una relación de muchos a muchos o de muchos a uno.
Las relaciones de uno a muchos propagan la clave principal de la entidad cuya cardinalidad es uno a la entidad de cardinalidad n.

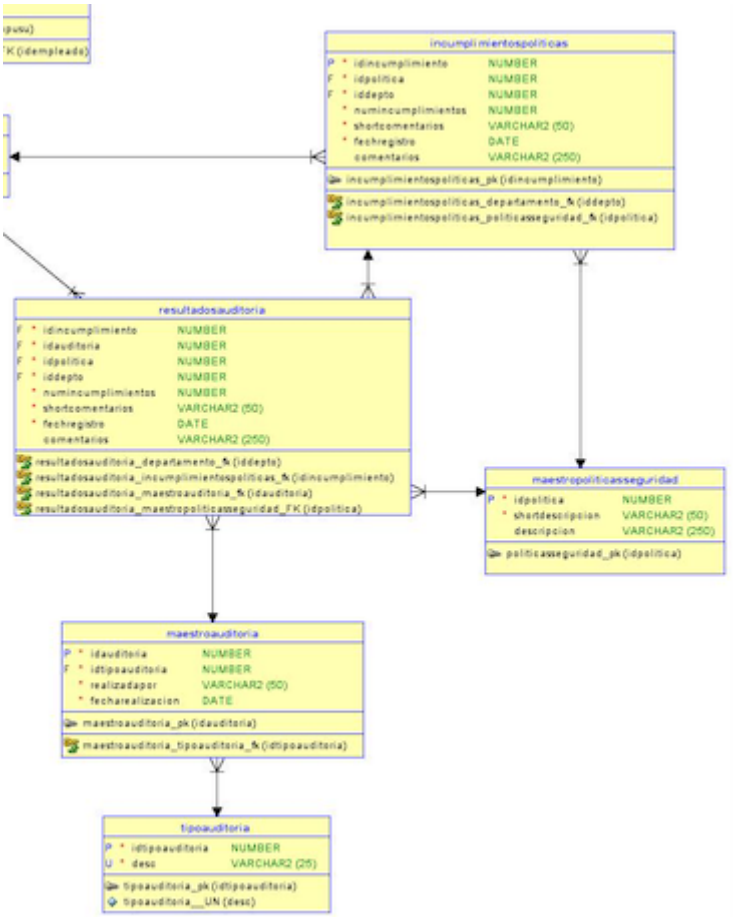
Para el **bloque de vulnerabilidades y mitigaciones** el diseño lógico sería:



Para el bloque de **empleados, departamentos y formaciones** el diagrama lógico sería:

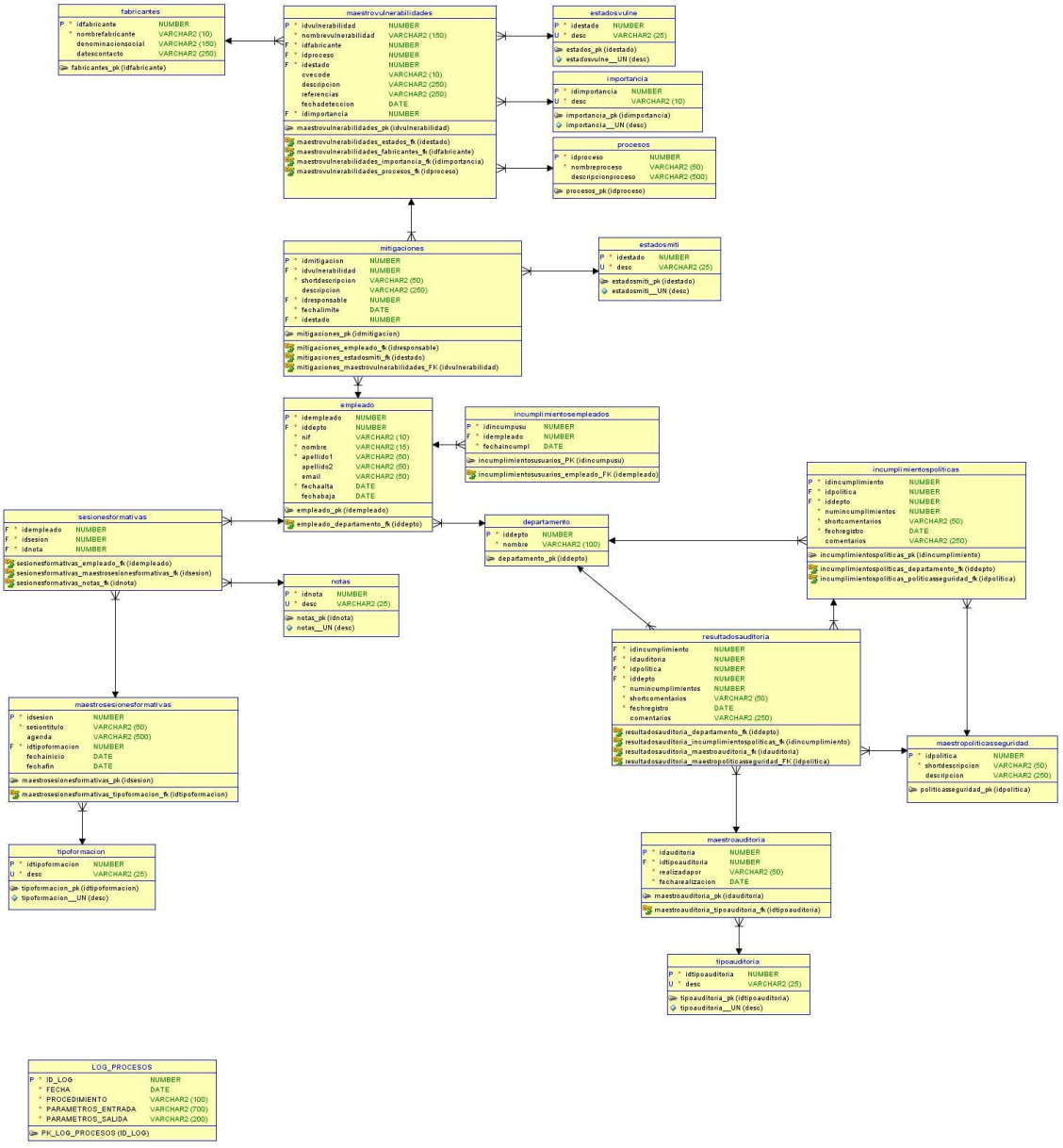


Para el bloque de **incumplimientos de políticas establecidas por la empresa,** este sería su modelo lógico:



Para el bloque del **sistema de log,** este sería su modelo lógico:

Agrupando todos los bloques, este sería su modelo lógico:



2.4 Diseño físico

Ahora llegamos a la parte del proyecto en la que tenemos que transformar nuestro modelo lógico a un modelo físico. Para ello tenemos que tener en cuenta las funcionalidades del sistema gestor de base de datos que estamos utilizando, en nuestro caso Oracle XE 21c.

Primero de todo, hemos de detenernos en una funcionalidad que hemos utilizado ampliamente en el desarrollo del modelo y que nos gustaría que explicar:
Los identificadores que hemos usado, secuenciales a todas luces, en el proyecto y que normalmente son primary key de las tablas, los hemos

conformado con una funcionalidad de Oracle disponible en la versión que hemos usado que son los identity column.
Su función es la de insertar un valor secuencial en una columna cada vez que se inserta un fila en una tabla, se suele utilizar para generar claves primarias. Su funcionalidad es muy similar a la utilización de secuencias pero su uso es más simple. No necesitaremos hacer un nextval de la secuencia y que este número se consiga cuando insertamos un nuevo dato usando un trigger. Nos simplificará enormemente la labor de codificación.

Como no podía ser de otra forma, la utilización de columnas IDENTITY en Oracle tiene una serie de restricciones que hemos de tener en cuenta:

- Solo puede haber una columna IDENTITY por tabla.
- Las columnas IDENTITY tienen que ser de tipo numérico, no se permiten tipos definidos por el usuario.
- Las columnas IDENTITY no pueden tener cláusula DEFAULT.
- Cuando utilizamos CREATE TABLE … AS SELECT la columna de la nueva tabla no será IDENTITY. Si queremos mantener la misma estructura tenemos que crear la tabla primero y luego realizar un INSERT AS SELECT.

Como podemos ver en el anexo en el que detallamos la instalación de Oracle XE 21c, no hemos utilizado el usuario dba para desplegar el modelo sino que hemos creado un usuario propio de la aplicación que disfruta de unos permisos más reducidos que el dba por cuestiones de seguridad. También hemos creado nuevas unidades físicas de almacenamiento para no crear ningún objeto de usuario en los tablespaces SYSTEM y SYSAUX que son propios del diccionario de datos.

A continuación pasaremos a detallar las tablas implicadas en nuestro modelo con sus correspondientes campos y restricciones según el diseño lógico que creamos en el apartado anterior.

Tabla: fabricantes				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idfabricante	NUMBER	PK	N	Identificador numérico del fabricante.
nombrefabricante	VARCHAR2(10)		N	nombre comercial del fabricante.
denominacionsocial	VARCHAR2(150)			datos con los que ha sido inscrita en el registro mercantil.
datoscontacto	VARCHAR2(250)			datos de contacto de la compañía, incluyendo en este campo cif, dirección, teléfono.

Tabla: maestrovulnerabilidades				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idvulnerabilidad	NUMBER	PK	N	Identificador de la vulnerabilidad.
nombrevulnarabilidad	VARCHAR2(150)		N	Nombre de la vulnerabilidad de forma descriptiva.
idfabricante	NUMBER	FK	N	Identificador del fabricante.
idproceso	NUMBER	FK	N	Identificador del proceso de negocio.
idestado	NUMBER	FK	N	Identificador del estado en el que se encuentra la vulnerabilidad.
cvecode	VARCHAR2(10)			CVE code s lo tiene.
descripcion	VARCHAR2(250)			descripción de la vulnerabilidad con detalle, qué es lo que afecta.
referencias	VARCHAR2(250)			Qué compañía la ha reportado o descubierto, etc. donde hay mas información sobre la misma.
fechadeteccion	DATE			Fecha de cuando se detectó.
idimportancia	NUMBER	FK	N	Identificador de la importancia.

Tabla: estadosvulne				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idesstado	NUMBER	PK	N	Identificador del estado.
desc	VARCHAR2(25)	U	N	Descriptivo.

Tabla: importancia				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idimportancia	NUMBER	PK	N	Identificador de importancia.
desc	VARCHAR2(10)	U	N	Descriptivo.

Tabla: procesos				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idproceso	NUMBER	PK	N	Identificador del proceso.
nombreproceso	VARCHAR2(50)		N	Nombre del proceso de negocio.
descripcionproceso	VARCHAR2(500)			Descripción del proceso de negocio.

Tabla: mitigaciones				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idmitigacion	NUMBER	PK	N	Identificador de la mitigación.
idvulnerabilidad	NUMBER	FK	N	Identificador de la vulnerabilidad.
shortdescripcion	VARCHAR2(50)		N	pequeña descripción de la mitigación.
descripcion	VARCHAR2(250)			larga descrición de la mitigación, detallada al máximo.
idresponsable	NUMBER	FK	N	Identificador del responsable de esta mitigación.
fechalimite	DATE		N	Fecha límite en la que la mitigación se ha de resolver.
idestado	NUMBER	FK	N	Identificador del estado en el que se encuentra la mitigación.

Tabla: estadosmiti				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idestado	NUMBER	PK	N	Identificador del estado.
desc	VARCHAR2(25)	U	N	Descriptivo.

Tabla: empleado				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idempleado	NUMBER	PK	N	Identificador empleado.
iddepto	NUMBER	FK	N	Identificador departamento.
nif	VARCHAR2(10)		N	Número NIF.
nombre	VARCHAR2(15)		N	Nombre del empleado.
apellido1	VARCHAR2(50)		N	Apellido del empleado.
apellido2	VARCHAR2(50)			Apellido segundo del empleado si tiene.
email	VARCHAR2(50)			Dirección de email.
fechaalta	DATE		N	Fecha de alta del empleado e la empresa.
fechabaja	DATE			Fecha de baja del empleado en la empresa.

Tabla: incumplimientosempleados				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idincumpusu	NUMBER	PK	N	Identificador de incumplimiento asociado a usuario.
idempleado	NUMBER	FK	N	Identificador de empleado.
fechaincumpl	DATE		N	Fecha del incumplimiento.

Tabla: departamento				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
iddepto	NUMBER	PK	N	Identificador del departamento.
nombre	VARCHAR2(100)		N	Nombre del departamento.

Tabla: incumplimientospoliticas				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idincumplimiento	NUMBER	PK	N	Identificador del incumplimiento.
idpolitica	NUMBER	FK	N	Identificador de política.
iddepto	NUMBER		N	Identificador de departamento.
numincumplimientos	NUMBER		N	Número de incumplimientos de esa política por parte de ese departamento.
shortcomentarios	VARCHAR2(50)		N	Comentarios breves sobre el incumplimiento.
fechregistro	DATE		N	Fecha de registro.
comentarios	VARCHAR2(250)		N	Ampliación de comentarios sobre el incumplimiento.

Tabla: maestropoliticasseguridad				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idpolitica	NUMBER	PK	N	Identificador de la política.
shortdescripcion	VARCHAR2(50)		N	Pequeña descripción de la política.
descripcion	VARCHAR2(250)			Ampliación de la descripción de la política.

Tabla: resultadosauditoria				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idincumplimiento	NUMBER	FK	N	Identificador del incumplimiento.
idauditoria	NUMBER	FK	N	Identificador de la auditoría.
idpolitica	NUMBER	FK	N	Identificador de la política.
iddepto	NUMBER	FK	N	Identificador del departamento.
numincumplimientos	NUMBER		N	Número de incumplimientos de dicha política por ese departamento.
shortcomentarios	VARCHAR2(50)		N	Comentarios cortos.

fechregistro	DATE		N	Fecha de registro.
comentarios	VARCHAR2(250)			Ampliación de los comentarios si aplica

Tabla: maestroauditoria				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idauditoria	NUMBER	PK	N	Identificador de auditoría.
idtipoauditoria	NUMBER	FK	N	Identificador del tipo de auditoría.
realizadapor	VARCHAR2(50)		N	Quién realizó la auditoría.
fecharealizacion	DATE		N	Fecha de la realización de la auditoría.

Tabla: tipoauditoria				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idtipoauditoria	NUMBER	PK	N	Identificador de la auditoría.
desc	VARCHAR2(25)	U	N	Descriptivo.

Tabla: sesionesformativas				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idempleado	NUMBER	FK	N	Identificador del empleado.
idsesion	NUMBER	FK	N	Identificador de la sesión.
idnota	NUMBER	FK	N	Identificación de la nota.

Tabla: maestrosesionesformativas				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idsesion	NUMBER	PK	N	Identificador de la sesión formativa.
sesiontitulo	VARCHAR2(50)		N	Título que tiene la sesión formativa.
agenda	VARCHAR2(500)			Agenda de la que consta la sesión formativa.
idtipoformacion	NUMBER	FK	N	Identificador del tipo de formación.
fechainicio	DATE			Fecha de inicio de la formación.
fechafin	DATE			Fecha fin de la formación.

Tabla: notas				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idnota	NUMBER	PK	N	Identificador de nota.
desc	VARCHAR2(25)	U	N	Descriptivo.

Tabla: tipoformacion				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
idtipoformacion	NUMBER	PK	N	Identificador de la formación.
desc	VARCHAR2(25)	U	N	Descriptivo.

Tabla:				
Campo	**Tipo**	**Clave**	**Nulo**	**Descripción**
id_log	NUMBER	PK	N	Identificador del log.
fecha	DATE		N	Fecha del registro del evento en el log.
procedimiento	VARCHAR2(100)		N	Nombre del procedimiento al que corresponde el registro.

parametros_entrada	VARCHAR2(700)		N	Qué parámetros de entrada llegaron.
parametros_salida	VARCHAR2(200)		N	Qué parámetros de salida recibimos.

2.5 Diseño procedimientos almacenados

En el enunciado del TFG se explicita:
“Toda la gestión y acceso a la información se hará mediante procedimientos de BD, siendo ésta la única manera de acceder. Por tanto, no será necesario implementar ninguna interfaz de usuario con un lenguaje de programación de alto nivel. A nivel de procedimientos, se deberá implementar y describir con detalle los procedimientos de ABM (Alta + Baja + Modificación) de todas las entidades (o clases) que se consideren relevantes.”

Por lo que hemos creado un paquete que será el encargado de realizar las inserciones, actualizaciones y borrados de datos en las tablas, este paquete se llamará PCK_ABM.

Para la resolución de las diferentes preguntas planteadas, no hemos creído necesaria la necesidad de creación de procedures o funciones, sino que hemos hecho uso de plsql dinámico que nos permite de una forma fácil montar cursores, usar variables, anidar bloques de código, etc. permitiendo dar solución a las preguntas planteadas de una forma simple, eso sí usaremos en todo momento sql estático sin funciones de agrupación tal como demanda el enunciado.

Procedimiento: ALTA_PROCESOS_NEGOCIO	
Descripción:	Dá de alta un nuevo proceso de negocio.
Parámetros de entrada	pnombreproceso IN VARCHAR2 pdescripcionproceso IN VARCHAR2
¿Qué hace?	Comprobamos que: ● el proceso no se haya insertado previamente. ● que el nombre del proceso no sea nulo. Si todo es correcto insertará el registro en la tabla procesos. El identificador del proceso es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK

	ERROR: El nombre del proceso no puede ser nulo. ERROR: El nombre del proceso ya está dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_PROCESOS_NEGOCIO	
Descripción:	Dá de baja un proceso de negocio.
Parámetros de entrada	pnombreproceso IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el nombre del proceso no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla procesos. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El nombre del proceso no puede ser nulo. ERROR: El nombre del proceso no existe. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_PROCESOS_NEGOCIO	
Descripción:	Modifica un proceso de negocio.
Parámetros de entrada	pnombreproceso IN VARCHAR2, pdescripcionproceso IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el nombre del proceso no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto modificará el registro en la tabla procesos. De existir FK se validará el dato a introducir para ver si existe registro padre.

Salidas esperadas	OK ERROR: El código del proceso no puede ser nulo. ERROR: El nombre de proceso no existe. ERROR: SQLCODE When others.

Procedimiento: ALTA_FABRICANTE	
Descripción:	Dá de alta un nuevo fabricante.
Parámetros de entrada	pnombrefabricante IN VARCHAR2, pdenominacion IN VARCHAR2, pdatoscontacto IN VARCHAR2
¿Qué hace?	Comprobamos que: ● el fabricante no se haya insertado previamente. ● que el nombre del fabricante no sea nulo. Si todo es correcto insertará el registro en la tabla fabricantes. El identificador del fabricante es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El nombre del fabricante no puede ser nulo. ERROR: El nombre del fabricante ya está dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_FABRICANTE	
Descripción:	Dá de baja un fabricante.
Parámetros de entrada	pnombrefabricante IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el nombre del fabricante no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla fabricantes. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo,

	teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El nombre del fabricante no puede ser nulo. ERROR: El nombre del fabricante no existe. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_FABRICANTE	
Descripción:	Modifica un fabricante.
Parámetros de entrada	pnombrefabricante IN VARCHAR2, pdenominacion IN VARCHAR2, pdatoscontacto IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el nombre del fabricante no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto modificará el registro en la tabla fabricantes. De existir FK se validará el dato a introducir para ver si existe registro padre.
Salidas esperadas	OK ERROR: El nombre del fabricante no puede ser nulo. ERROR: El nombre del fabricante no existe ERROR: SQLCODE When others.

Procedimiento: ALTA_DPTO	
Descripción:	Dá de alta un nuevo departamento.
Parámetros de entrada	pnombre IN VARCHAR2
¿Qué hace?	Comprobamos que: ● el departamento no se haya insertado previamente. ● que el nombre del departamento no sea nulo. Si todo es correcto insertará el registro en la tabla departamento. El identificador del departamento es generado con un campo

	identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El nombre del dpto no puede ser nulo. ERROR: El nombre del dpto ya está dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_DPTO	
Descripción:	Dá de baja un departamento.
Parámetros de entrada	pnombre IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el nombre del departamento no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla departamento. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El nombre del dpto no puede ser nulo. ERROR: El nombre del dpto no existe. ERROR: SQLCODE When others.

Procedimiento: ALTA_EMPLEADO	
Descripción:	Dá de alta un nuevo empleado.
Parámetros de entrada	pNIF IN VARCHAR2, pnombre IN VARCHAR2, papellido1 IN VARCHAR2, papellido2 IN VARCHAR2, piddepto IN NUMBER, pfechaalta IN DATE,

	pfechabaja IN DATE, pemail IN VARCHAR2
¿Qué hace?	Comprobamos que: ● el empleado no se haya insertado previamente. ● que el NIF del empleado no sea nulo. ● que el nombre del empleado no sea nulo. ● que el primer apellido del empleado no sea nulo. ● que el departamento al que pertenece el empleado no sea nulo. ● La fecha de alta del empleado en la compañía no puede ser nula. Si todo es correcto insertará el registro en la tabla empleado. El identificador del empleado es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El NIF del empleado no puede ser nulo. ERROR: El nombre del empleado no puede ser nulo. ERROR: El apellido1 del empleado no puede ser nulo. ERROR: El depto del empleado no puede ser nulo. ERROR: La fecha de alta del empleado no puede ser nulo. ERROR: El empleado ya está dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_EMPLEADO	
Descripción:	Dá de baja un empleado.
Parámetros de entrada	pNIF IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el NIF del empleado no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la

	tabla empleado. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El NIF del empleado no puede ser nulo. ERROR: El NIF del empleado no existe. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_EMPLEADO	
Descripción:	Modifica los datos de un empleado.
Parámetros de entrada	pNIF IN VARCHAR2, pnombre IN VARCHAR2, papellido1 IN VARCHAR2, papellido2 IN VARCHAR2, piddepto IN NUMBER, pfechaalta IN DATE, pfechabaja IN DATE, pemail IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el registro existe previamente en la tabla. ● que el NIF del empleado no sea nulo. ● que el nombre del empleado no sea nulo. ● que el primer apellido del empleado no sea nulo. ● que el departamento al que pertenece el empleado no sea nulo. ● La fecha de alta del empleado en la compañía no puede ser nula. Recordemos que algún campo de esta tabla es FK por lo que validará el dato que van a meter buscando sus padres, de no existir saltará el error en WHENOTHERS.
Salidas esperadas	OK ERROR: El NIF del empleado no puede ser nulo. ERROR: El nombre del empleado no puede

	ser nulo. ERROR: El apellido1 del empleado no puede ser nulo. ERROR: El depto del empleado no puede ser nulo. ERROR: La fecha de alta del empleado no puede ser nulo. ERROR: El empleado ya está dado de alta. ERROR: SQLCODE When others.

Procedimiento: ALTA_SESIONESFORMATIVAS	
Descripción:	Dá de alta una nueva sesión formativa.
Parámetros de entrada	pidempleado IN NUMBER, pidsesion IN NUMBER
¿Qué hace?	Comprobamos que: . ● que el id del empleado no sea nulo. ● que el id de la sesión formativa no sea nulo. ● id de la nota nunca será nulo ya que de no meter valor ponemos uno por defecto (valor 1). Si todo es correcto insertará el registro en la tabla sesionesformativas. Recordemos que todos los campos de esta tabla son FK por lo que validará el dato que van a meter buscando sus padres.
Salidas esperadas	OK ERROR: El ID del empleado no puede ser nulo. ERROR: El ID de la sesión formativa no puede ser nulo. ERROR: SQLCODE When others.

Procedimiento: BAJA_SESIONESFORMATIVAS	
Descripción:	Borra una sesión formativa.
Parámetros de entrada	pidempleado IN NUMBER, pidsesion IN NUMBER
¿Qué hace?	Comprobamos que: ● que el id del empleado no sea nulo.

	• que el id de la sesión formativa no sea nulo. • que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla sesionesformativas. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El ID del empleado no puede ser nulo. ERROR: El ID de la sesión formativa no puede ser nulo. ERROR: El ID del empleado no existe en la sesión formativa indicada. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_SESIONESFORMATIVAS	
Descripción:	Modifica los datos de una sesión formativa.
Parámetros de entrada	pidempleado IN NUMBER, pidsesion IN NUMBER, pidnota IN NUMBER
¿Qué hace?	Comprobamos que: • que el id del empleado no sea nulo. • que el id de la sesión formativa no sea nulo. • que el id de la nota no puede ser nulo. • que el registro existe previamente en la tabla. Recordemos que todos los campos de esta tabla es FK por lo que validará los datos que se van a meter buscando sus padres, de no existir saltará el error en WHENOTHERS.
Salidas esperadas	OK ERROR: El ID del empleado no puede ser nulo. ERROR: El ID de la sesión formativa no

	puede ser nulo. ERROR: El ID de la nota no puede ser nulo. ERROR: SQLCODE When others.

Procedimiento: ALTA_MAESTROSESIONESFORMATIVAS	
Descripción:	Introducimos la información relativa a la nueva sesión formativa.
Parámetros de entrada	psesiontitulo IN VARCHAR2, pagenda IN VARCHAR2, pidtipoformacion IN NUMBER, pfechainicio IN DATE, pfechafin IN DATE
¿Qué hace?	Comprobamos que: ● la sesión formativa no se haya insertado previamente. ● que el título de la sesión no sea nulo. ● que el id del tipo de formación no sea nulo. Si todo es correcto insertará el registro en la tabla maestrosesionesformativas. El identificador de la sesión formativa es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El título de la sesión no puede ser nulo. ERROR: El Id del tipo de formación de la sesión no puede ser nulo. ERROR: El titulo de la sesion ya esta dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_MAESTROSESIONESFORMATIVAS	
Descripción:	Eliminamos la información que tengamos de una sesión formativa.
Parámetros de entrada	psesiontitulo IN VARCHAR2

¿Qué hace?	Comprobamos que: ● que el título de la sesión formativa no sea nulo. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla maestrosesionesformativas. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: El título de la sesión formativa no puede ser nulo. ERROR: La sesión formativa no existe. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_MAESTROSESIONESFORMATIVAS	
Descripción:	Modificamos la información relativa a una sesión formativa.
Parámetros de entrada	psesiontitulo IN VARCHAR2, pagenda IN VARCHAR2, pidtipoformacion IN NUMBER, pfechainicio IN DATE, pfechafin IN DATE
¿Qué hace?	Comprobamos que: ● que el título de la sesión no sea nulo. ● que el tipo de formación no sea nulo. ● que el registro existe previamente en la tabla. Recordemos que algún campo de esta tabla es FK por lo que validará los datos que se van a meter buscando sus padres, de no existir saltará el error en WHENOTHERS.
Salidas esperadas	OK ERROR: El título de la sesión no puede ser nulo. ERROR: El tipo de formación no puede ser nulo. ERROR: El título de la sesión no existe.

	ERROR: SQLCODE When others.

Procedimiento: ALTA_DATOS_LOOKUP	
Descripción:	Dá de alta un registro en una tabla de búsqueda.
Parámetros de entrada	ptable IN VARCHAR2, pdatolookup IN VARCHAR2
¿Qué hace?	Comprobamos que: ● El nombre de la tabla de lookup no sea nulo. ● que el dato lookup a insertar no sea nulo. ● que la tabla de lookup en la que vamos a insertar, exista. ● Que no haya registro repetido, lo cual solucionaremos con una constraint unique que hará que el error salte por whenothers Si todo es correcto insertará el registro en la tabla de lookup que le pasemos por parámetro. El identificador del registro es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El nombre de la tabla de lookup no puede ser nulo. ERROR: El dato a insertar en la tabla de lookup no puede ser nulo. ERROR: La tabla de lookup no existe. ERROR: SQLCODE When others.

Procedimiento: ALTA_VULNERABILIDAD	
Descripción:	Dá de alta una vulnerabilidad.
Parámetros de entrada	pnombre IN VARCHAR2, pidfabricante IN NUMBER, pidproceso IN NUMBER, pidestado IN NUMBER,

	pcvecode IN VARCHAR2, pdescripcion IN VARCHAR2, preferencias IN VARCHAR2, pfechadeteccion IN DATE, pidimportancia IN NUMBER
¿Qué hace?	Comprobamos que: ● que el nombre de la vulnerabilidad no sea nulo. ● que el id del proceso no sea nulo. ● que el id del fabricante no sea nulo. ● que el id del estado no sea nulo. ● que el id de la importancia no sea nulo. ● que el nombre de la vulnerabilidad no se encuentre ya registrado en la tabla y por tanto no haya valores repetidos para una misma vulnerabilidad. Si todo es correcto insertará el registro en la tabla maestrovulnerabilidades. El identificador del registro es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger. Encontramos también claves FK en la tabla que validará el dato que hemos introducido y que de no encontrar su valor padre harán saltar un error oracle en whenothers.
Salidas esperadas	OK ERROR: El nombre de la vulnerabilidad no puede ser nulo. ERROR: El Id del proceso de empresa no puede ser nulo. ERROR: El Id fabricante no puede ser nulo. ERROR: El ID de estado no puede ser nulo. ERROR: El Id de la importancia no puede ser nulo. ERROR: Esta vulnerabilidad ya se ha dado de alta. ERROR: SQLCODE When others.

Procedimiento: ALTA_MITIGACION	
Descripción:	dá de alta una mitigación de una vulnerabilidad detectada.
Parámetros de entrada	pidvulnerabilidad IN NUMBER, pshortdescripcion IN VARCHAR2, pdescripcion IN VARCHAR2, pidresponsable IN NUMBER, pfechalimite IN DATE, pidestado IN NUMBER
¿Qué hace?	Comprobamos que: ● que el código de la vulnerabilidad no sea nulo. ● que la descripción de la mitigación no sea nula. ● que el id responsable de la mitigación no sea nulo. ● que la fecha máxima de la mitigación no sea nula. ● que el id del estado no sea nulo. Si todo es correcto insertará el registro en la tabla mitigaciones. El identificador del registro es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger. Encontramos también claves FK en la tabla que validará el dato que hemos introducido y que de no encontrar su valor padre harán saltar un error oracle en whenothers
Salidas esperadas	OK ERROR: El código de la vulnerabilidad no puede ser nulo. ERROR: La descripción de la mitigación no puede ser nulo. ERROR: El ID del responsable no puede ser nulo. ERROR: La fecha límite de mitigación no puede ser nulo. ERROR: El ID del estado no puede ser nulo. ERROR: La mitigación ya ha sido dada de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_MITIGACION	
Descripción:	Dá de baja una mitigación.
Parámetros de entrada	pshortdescripcion IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que la pequeña descripción de la mitigación no sea nula. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla mitigaciones. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	OK ERROR: La short description no puede ser nula. ERROR: No encontrada la short description no existe. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_MITIGACION	
Descripción:	Modifica una mitigación dada de alta anteriormente.
Parámetros de entrada	pidvulnerabilidad IN NUMBER, pshortdescripcion IN VARCHAR2, pdescripcion IN VARCHAR2, pidresponsable IN NUMBER, pfechalimite IN DATE, pidestado IN NUMBER
¿Qué hace?	Comprobamos que: ● que el código de la mitigación no sea nulo. ● que la descripción de la mitigación no sea nula. ● que el id del responsable de resolver la mitigación no sea nulo.

	• que la fecha límite de resolución de la mitigación no sea nula. • que el id del estado de la mitigación no sea nulo. • que el registro existe previamente en la tabla. Recordemos que algún campo de esta tabla es FK por lo que validará los datos que se van a meter buscando sus padres, de no existir saltará el error en WHENOTHERS.
Salidas esperadas	OK ERROR: El código de la vulnerabilidad no puede ser nulo. ERROR: La descripción de la mitigación no puede ser nulo. ERROR: El ID del responsable no puede ser nulo. ERROR: La fecha límite de mitigación no puede ser nulo. ERROR: El ID del estado no puede ser nulo. ERROR: SQLCODE When others.

Procedimiento: ALTA_MAESTROPOLITICAS	
Descripción:	Dá de alta en el sistema una política de empresa.
Parámetros de entrada	pshortdescripcion IN VARCHAR2, pdescripcion IN VARCHAR2
¿Qué hace?	Comprobamos que: • que la descripción de la política no sea nula. • que la política no haya sido dada de alta anteriormente. Si todo es correcto insertará el registro en la tabla maestropoliticasseguridad. El identificador del registro es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK

	ERROR: La descripción de la política no puede ser nulo. ERROR: La política ya ha sido dada de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_MAESTROPOLITICAS	
Descripción:	Dá de baja del sistema una política.
Parámetros de entrada	pshortdescripcion IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que la pequeña descripción de la politica no sea nula. ● que el registro existe previamente en la tabla. Si todo es correcto borrará el registro en la tabla maestropoliticasseguridad. La relación con otras tablas está basada una relación fuerte, no podremos borrar el registro si existen otras dependencias (PK/FK) sobre el mismo, teniendo que borrar los registros hijos antes de poder borrar el padre.
Salidas esperadas	ERROR: La descripción de la política no puede ser nulo. ERROR: SQLCODE When others.

Procedimiento: MODIFICACION_MAESTROPOLITICAS	
Descripción:	Modifica una política.
Parámetros de entrada	pshortdescripcion IN VARCHAR2, pdescripcion IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que la pequeña descripción de la política no sea nula. ● que el registro existe previamente en la tabla.
Salidas esperadas	OK ERROR: La short description no puede ser nula.

	ERROR: SQLCODE When others.

Procedimiento: ALTA_INCUMPLIMIENTOSPOLITICAS	
Descripción:	Dá de alta un incumplimiento de política.
Parámetros de entrada	pidpolitica IN NUMBER, piddepto IN NUMBER, pnumincumplimientos IN NUMBER, pshortcomentarios IN VARCHAR2, pcomentarios IN VARCHAR2
¿Qué hace?	Comprobamos que: ● que el código de la política no sea nulo. ● que el dpto no puede ser nulo. ● que el número de incumplimientos no ha de ser nulo. ● que la descripción corta de la política no puede ser nula. ● que el incumplimiento no haya sido dado de alta anteriormente. Si todo es correcto insertará el registro en la tabla incumplimientospoliticas. El identificador del registro es generado con un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El código de la política no puede ser nulo. ERROR: El departamento asociado a la política no puede ser nulo. ERROR: El número de incumplimientos de la política no puede ser nulo. ERROR: Descripción corta de la política no puede ser nulo. ERROR: El incumplimiento de la politica ya fue dado de alta. ERROR: SQLCODE When others.

Procedimiento: BAJA_INCUMPLIMIENTOSPOLITICAS	
Descripción:	Dá de baja un incumplimiento de política.
Parámetros de entrada	pidpolitica IN NUMBER, pshortcomentarios IN VARCHAR2,
¿Qué hace?	Comprobamos que: ● que el código de la política no sea nulo. ● que la descripción corta de la política no puede ser nula. ● que el incumplimiento no haya sido dado de alta anteriormente. Si todo es correcto borrará el registro en la tabla incumplimientospoliticas.
Salidas esperadas	OK ERROR: El código de la política no puede ser nulo. ERROR: Descripción corta de la política no puede ser nulo. ERROR: El incumplimiento de la politica ya fue dado de alta. ERROR: SQLCODE When others.

Procedimiento: ALTA_INCUMPLIMIENTOSEMPLEADOS	
Descripción:	Dá de alta un incumplimiento realizado por un empleado.
Parámetros de entrada	pidempleado IN NUMBER, pfechaincumpl IN DATE
¿Qué hace?	Comprobamos que: ● que el empleado que incumple la política está informado. ● que la fecha en la que incumplió el empleado la política no es nula. ● que no hemos insertado previamente este mismo registro. Si todo es correcto insertará el registro en la tabla incumplimientosempleados. El identificador del registro es generado con

	un campo identity por lo que no nos tenemos que preocupar de generar ni secuencia ni trigger.
Salidas esperadas	OK ERROR: El id del empleado que incumple la política no puede ser nulo. ERROR: La fecha en el que el empleado incumplió la política no puede ser nulo. ERROR: El incumplimiento por este empleado ya fue dado de alta.

Procedimiento: BAJA_INCUMPLIMIENTOSEMPLEADOS	
Descripción:	Dá de bajaun incumplimiento realizado por un empleado.
Parámetros de entrada	pidempleado IN NUMBER, pfechaincumpl IN DATE
¿Qué hace?	Comprobamos que: ● que el empleado que incumple la política está informado. ● que la fecha en la que incumplió el empleado la política no es nula. ● que no hemos insertado previamente este mismo registro. Si todo es correcto borrará el registro en la tabla incumplimientosempleados.
Salidas esperadas	OK ERROR: El id del empleado que incumple la política no puede ser nulo. ERROR: La fecha en el que el empleado incumplió la política no puede ser nulo. ERROR: El incumplimiento por este empleado ya fue dado de alta.

Procedimiento: CREA_AUDITORIA	
Descripción:	Crea una auditoria de la informacion que tenemos en ese momento en la tabla maestro especificando el tipo de auditoria

	que vamos a hace y el periodo a estudiar en la misma.
Parámetros de entrada	prealizadapor VARCHAR ptipoauditoria NUMBER pfechainicial DATE pfechafinal DATE
¿Qué hace?	Comprobamos que: ● esté informado la persona que va a hacer la auditoria ● que el tipo de auditoría esté especificado. ● que pongamos una fecha de inicio. ● que indiquemos una fecha de fin. Si todo es correcto insertará los registros de incumplimientospolíticas a resultadosauditoría con los criterios indicados.
Salidas esperadas	OK ERROR: Quien realizo la auditoria no puede ser nulo. ERROR: tipo de auditoria no puede ser nulo. ERROR: Ninguna de las fechas del periodo a auditar no puede ser nula.

2.6 Resolución consultas planteadas

Como comentamos anteriormente no podemos usar funciones de agrupación ni tampoco el uso de vistas, lo que simplificaría enormemente la codificación de la resolución a las preguntas planteadas. Los diferentes resultados de las consultas se definan en tiempo constante 1, bajo esas premisas:

	Consulta 1.-
Descripción	Departamento que, en un año concreto, tiene un número mayor de incumplimientos de seguridad registrados en la BD.
Parámetros entrada	Año para el que queremos obtener la información.
¿qué hace?	Cargamos en un cursor los datos de todos los departamentos que tuvieron incumplimientos el año

	que se nos especifica. Para cada departamento que hemos sacado, sumamos el número de incumplimientos por políticas, obteniendo el número de incumplimientos de ese departamento en todas las políticas. Usamos dos variables de apoyo para almacenar el número mayor de incumplimientos e ir comparando con el siguiente departamento. Al final del loop habré obtenido el departamento que mayor número de incumplimientos ha tenido.
Salida	El departamento '||**nombbredep**||' en '||**&anyo**||' tuvo el máximo número de incumplimientos... '||**maxincumplimientos**

	Consulta 2.-
Descripción	Proceso de gestión interno que, teniendo en cuenta toda la información de que se dispone en la BD, ha tenido un mayor número de vulnerabilidades detectadas.
Parámetros entrada	
¿qué hace?	Cargamos en un cursor los id de todos los procesos, anidamos dentro de este otro que sacará las vulnerabilidades, como no puedo usar un count he de usar un contador auxiliar para obtener el número. Grabaremos en dos variables los valores que vayamos obteniendo guardando en las mismas el valor del id process y del máximo de vulnerabilidades que se encontraron.
Salida	'El proceso con id '||**procmaxid**||' tiene '||**vulnerabmax**||' vulnerabilidades.'

	Consulta 3.-
Descripción	Top5 de usuarios por número de incumplimientos asociados directamente a ellos, o a su departamento, durante el año en curso.
Parámetros entrada	
¿qué hace?	Obtenemos el listado de incumplimientos ordenado, eliminando los empleados que tengan 0. Por un lado sacamos los del usuario y por otro los del equipo, usamos una tabla auxiliar para almacenar cada incumplimiento y poder totalizarlo.

Salida	En '||c_resultado.rownum||' posicion... '||c_resultado.nombre||' '||c_resultado.apellido1||' con '||c_resultado.suma||' incumplimientos.

	Consulta 4.-
Descripción	Porcentaje de vulnerabilidades que, en el momento de ejecutar la consulta, están totalmente mitigadas.
Parámetros entrada	
¿qué hace?	Cargo en un cursor todas las vulnerabilidades y con un contador obtenemos cuántas son. Mediante otro cursor, obtenemos los id de las vulnerabilidades y anido otro cursor que sacará los datos que queremos viendo las mitigaciones totalmente acabadas, que vamos cargando en un contador. Cuando hemos terminado el loop, procedemos a imprimir los datos obtenidos por pantalla.
Salida	'Un '|| **round(completas/vulnerabtotal*100, 2)** ||'% de las vulnerabilidades totalmente mitigadas.'

	Consulta 5.-
Descripción	Número total de acciones de mitigación que, en el momento de ejecutar la consulta, no están totalmente acabadas.
Parámetros entrada	
¿qué hace?	Construimos la query que satisface los requisitos de la pregunta y cargamos en un cursor los valores que obtenemos con el único fin de usar una variable contador en la que iremos sumando 1 por cada registro, obteniendo así el número de mitigaciones solicitadas.
Salida	'Hay '||**contador**||' mitigaciones NO acabadas completamente.'

	Consulta 6.-
Descripción	Política de seguridad que, en el momento de ejecutar la consulta, ha tenido más incumplimientos (teniendo en cuenta todos los departamentos de la empresa)

Parámetros entrada	
¿qué hace?	Montamos un primer cursor en el que iremos cargando los id de las políticas que hay en ese momento en bbdd, para cada uno de esos id vamos a construir un segundo cursor cuya query satisface los valores que vamos buscando. Iniciamos el loop y nos ayudamos de unas variables intermedias en las que iremos guardando el id de la política de seguridad que más incumplimientos ha tenido.
Salida	'La política de seguridad que ha tenido más incumplimientos es la que tiene id. '||**idpoliticamax**||' con descripcion corta '||**shordescripcionmax**||' y tiene '||**nincumplimientosmax**||' incumplimientos.'

	Consulta 7.-
Descripción	Dado un determinado departamento de la empresa, y teniendo en cuenta el momento de ejecutar la consulta, porcentaje de usuarios del departamento que no han acabado todas las formaciones de seguridad asignadas.
Parámetros entrada	Nombre del departamento.
¿qué hace?	Obtenemos el número de empleados que pertenecen al departamento que nos llega por el parámetro de entrada. A continuación, obtenemos el número de empleados distintos del departamento que tienen formaciones en estados ('SIN CALIFICAR','NO PRESENTADO'), pudiendo sacar así el porcentaje solicitado.
Salida	'El '||**round(noterminadas/empleadosdpto*100,2)**||'% es el porcentaje de los empleados que no han terminado las acciones formativas asignadas para el departamento'

	Consulta 8.-
Descripción	Porcentaje de usuarios de la empresa que, en el año en curso, no tienen ningún incumplimiento asignado.
Parámetros entrada	
¿qué hace?	Mediante el uso de un cursor que saca todos los empleados y el uso de una variable temporal contador,

	sacamos el número total de empleados. Montamos otro cursor en el que sacaremos los empleados que tuvieron incumplimientos durante el año en curso. Pudiendo obtener de esta forma el porcentaje solicitado.
Salida	'Por tanto, un '||**((numempleados-numempleadosconincumplimientos)/numempleados*100)**||'% no comenten incumplimientos en '||**extract(year from sysdate)**

	Consulta 9.-
Descripción	Teniendo en cuenta todas las auditorías externas realizadas, año en el cual se han detectado más incumplimientos (teniendo en cuenta sólo los detectados durante la auditoría).
Parámetros entrada	
¿qué hace?	Nos creamos una tabla temporal en la que iremos actualizando los registros de ese año que cumplen con el criterio indicado.
Salida	El anyo con mas incumplimientos en todas las auditorias externas registradas fue el:'||**c_resultados.anyo**||' con '||**c_resultados.suma**||' incumplimientos.

	Consulta 10.-
Descripción	Porcentaje de vulnerabilidades críticas que, en el momento de ejecutar la consulta, tienen alguna acción de mitigación abierta (que no esté en estado “acabada”).
Parámetros entrada	Montamos un cursor con el que sacaremos las vulnerabilidades críticas, como venimos haciendo mediante el uso de una variable contador obtenemos su número. Con otro cursor sacamos lid de las vulnerabilidades críticas y dentro de este cursor, montaremos otro para sacar es estado que tiene su resolución. Cargando en un contador las que están ‘ACABADAS’ obtenemos la solución a la pregunta planteada.
¿qué hace?	Sacamos el número total de vulnerabilidades, posteriormente con un contador sacamos las que son

	críticas. Cuando tenemos estos dos datos averiguamos el porcentaje.
Salida	'Un '|| **round((vulnerabtotal-completas)/vulnerabtotal*100, 2)** ||'% de las vulnerabilidades CRITICAS con alguna acción de mitigación abierta.'

	Consulta 11.-
Descripción	Teniendo en cuenta el último año (el anterior al año en curso), título de la sesión formativa telemática que ha tenido un porcentaje menor de participantes en total.
Parámetros entrada	
¿qué hace?	Sacamos la información que necesitamos de las sesiones que fueron vía telemática, con el filtro del año anterior. Nos valemos de una tabla auxiliar en la que iremos haciendo los updates de los empleados que tomaron la formación con el fin de no usar funciones de agrupamiento.
Salida	**titulominparticip**||' tiene un porcentaje de participación del '||**round((contadorminparticipantessesion/contadorparticipantestotal*100),2)**||'% de los inscritos a alguna sesión formativa.'

	Consulta 12.-
Descripción	Número de vulnerabilidades críticas detectadas internamente teniendo en cuenta todos los datos de que se dispone. Se consideran detectadas internamente si se detectan con posterioridad al análisis realizado al inicio del proyecto por la consultora externa.
Parámetros entrada	Fecha finalización análisis consultora externa.
¿qué hace?	Sacamos mediante el uso de un contador y de un cursor el número de vulnerabilidades posterior a la fecha en la que se fueron, una única query que cumple todos los parámetros requeridos por el enunciado.
Salida	**numvulnerabilidadesafterleave**||' Vulnerabilidades CRITICAS fueron encontradas internamente despues del analisis inicial realizado por la consultora externa,

	despues del '||**to_date('&fechaleave','DD/MM/YYYY')**

	Consulta 13.-
Descripción	En el momento de ejecutar la consulta, porcentaje de acciones de mitigación en el sistema que están en los estados “en proceso” o “en revisión”.
Parámetros entrada	
¿qué hace?	Montamos un cursor con un contador para sacar el número de mitigaciones, usando otro cursor sacamos las que se encuentran en estado 'EN PROCESO' o 'EN REVISION'. Con estos datos ya obtenemos el resultado solicitado.
Salida	'Lo cual supone que hay un '||**round((mitigacionesestado/mitigacionestotal*100),2)**||'% del total de mitigaciones en estado "EN PROCESO" o "EN REVISION".'

	Consulta 14.-
Descripción	Teniendo en cuenta todas las acciones de mitigación en estado “en proceso”, persona responsable con más acciones asignadas.
Parámetros entrada	
¿qué hace?	Mediante dos cursores, en el primero voy buscando para cada empleado cuantas acciones de mitigación tiene en proceso, con los resultados obtenidos salvo el resultado en unas variables temporales que almacenarán el que más tiene.
Salida	El empleado con Id. '||**maxidresp**||' '||**nombre**||' '||**apellido**||' tiene '||**maxacciones**||' acciones de remediacion "en proceso", es el que más tiene.

2.7 Pruebas

Una vez tenemos el código plsql compilado estaremos seguros que sintácticamente cumple con las estructura necesaria para su ejecución, pero no estaremos seguros de si el código que hemos desarrollado cumple con lo que nosotros pretendemos que haga. Es por esto, que necesitamos ejecutar distintas pruebas unitarias, para comprobar que lo que queríamos hacer

realmente se hace correctamente. Con un buen set de pruebas eliminaremos futuras aperturas de incidencias y aumentará la satisfacción del cliente, de aquí su importancia.

2.7.1 Pruebas unitarias PCK_ABM

Como recordaremos hemos creado un paquete a través del cual somos capaces de insertar, borrar, modificar los registros de las diferentes tablas que hemos creado. Por un lado, creamos un set de datos de carga inicial, que nos ha servido para codificar las distintas consultas que proporcionan la información que se nos solicita responder en el enunciado del TFG, en este set de datos llamado “CargaDatos.sql” no hemos comprobado la totalidad de la funcionalidad del paquete que hemos codificado, básicamente hemos comprobado los procedures de inserción. Las pruebas unitarias están pensadas para comprobar la funcionalidad de cada uno de los procedures que hemos codificado. Bajo el nombre “pruebas_unitarias_PCK_ABM.sql” encontraremos las diferentes pruebas que hemos realizado y abajo en este documento las salidas esperadas y si efectivamente se han producido.

Procedures PCK_ABM	Resultado	ok/ko?
ALTA_PROCESOS_NEGOCIO	Alta de un proceso de negocio que no existe, cumpliendo las validaciones.	ok
	Error cuando el nombre de proceso es nulo.	ok
	Error porque el proceso de negocio ya existe	ok
BAJA_PROCESOS_NEGOCIO	Baja de un proceso de negocio si existe, cumpliendo las validaciones.	ok
	Error si nombre de proceso null	ok
	Error si nombre de proceso de negocio no está en base de datos.	ok
MODIFICACION_PROCESOS_NEGOCIO	Modificación de un proceso de negocio que existe, cumpliendo validaciones	ok
	Error si nombre de proceso es null	ok
	Error si el nombre de proceso no existe en base de datos	ok
ALTA_FABRICANTE	Alta de un fabricante que no existe, cumpliendo las validaciones.	ok
	Error el nombre de fabricante es null	ok
	El nombre del fabricante ya existe en la bbdd	ok
MODIFICACION_FABRICANTE	Modificación de un fabricante que existe, cumpliendo validaciones	ok
	Error si nombre de fabricante is null	ok

	Error si nombre de fabricante a modificar no encontrado en bbdd	ok
BAJA_FABRICANTE	Borrado de fabricante, cumpliendo validaciones	ok
	Error si nombre de fabricante nulo.	ok
	Error si fabricante a borrar no está en bbdd.	ok
ALTA_DPTO	Alta de un departamento que no existe, cumpliendo validaciones.	ok
	Error si nombre dpto esnull.	ok
	Error si dpto ya existe	ok
BAJA_DPTO	Borrado de departamento si existe, cumplien validaciones.	ok
	Error si nombre del dpto es null.	ok
	Error si el departamento no existe.	ok
ALTA_EMPLEADO	Alta de un empleado que no existe, cumpliendo validaciones.	ok
	Error si el empleado ya existe.	ok
	Error si el NIF es null	ok
	Error si el nombre el null.	ok
	Error si el apellido1 es null	ok
	Error si el departamento asociado al usuario es nulo.	ok
	Error si la fecha de alta es nula.	ok
MODIFICACION_EMPLEADO	Modificación de un empleado que existe, cumpliendo las validaciones.	ok
	Error si el empleado no existe.	ok
	Error si el NIF es null	ok
	Error si el nombre el null.	ok
	Error si el apellido1 es null	ok
	Error si el departamento asociado al usuario es nulo.	ok
	Error si la fecha de alta es nula.	ok
BAJA_EMPLEADO	Borrado de empleado, cumpliendo validaciones.	ok
	Error si NIF nulo.	ok
	Error si el empleado no existe.	ok
ALTA_MAESTROSESIONESFORMATIVAS	Alta de una sesión formativa en el maestro que no existe, cumpliendo validaciones.	ok
	Error si el título de la sesión es nulo.	ok
	Error si el id es nulo.	ok

	Error si ya hay otra sesión formativa con el mismo título.	ok
MODIFICACION_MAESTROSESIONESFORMATIVAS	Modificación del maestro de una sesión formativa que existe, cumpliendo las validaciones.	ok
	Error si título de la sesión formativa vacío.	ok
	Error si el tipo de formación es nulo.	ok
	Error si la sesión formativa no existe.	ok
BAJA_MAESTROSESIONESFORMATIVAS	Borrado del maestro de sesión formativa si existe, cumpliendo validaciones.	ok
	Error si el título está vacío.	ok
	Error si no encuentra la sesión formativa.	ok
ALTA_SESIONESFORMATIVAS	Alta de una sesión formativa a la que asistirá un empleado, cumpliendo validaciones.	ok
	Error, el id de empleado es nulo.	ok
	Error, el id de la sesión formativa es nulo.	ok
MODIFICACION_SESIONESFORMATIVAS	Modifica las sesiones informativas si cumplen las validaciones.	ok
	Error, el id de empleado es nulo.	ok
	Error el id de la sesión formativa es nulo.	ok
	Error el id de la nota es nulo.	ok
BAJA_SESIONESFORMATIVAS	Borrado de sesión formativa a la que se ha inscrito el empleado, respetando validaciones.	ok
	Error si el título está vacío.	ok
	Error si no encuentra la sesión formativa.	ok
ALTA_MITIGACIONES	Alta de una mitigación si no existe, cumpliendo validaciones.	ok
	Error, el código de la vulnerabilidad es null	ok
	Error, la descripción de la mitigación es null	ok
	Error, el id del responsable de la mitigación no puede ser nulo.	ok
	Error, la fecha límite de la mitigación no puede ser nulo.	ok
	Error, el id de estado no puede ser nulo.	ok
	Error, la short description de mitigación ya existe.	ok
MODIFICACION_MITIGACION	Modifica las mitigaciones si existen y si cumplen las validaciones.	ok
	Error, el código de la vulnerabilidad es null	ok
	Error, la descripción de la mitigaciones null	ok
	Error, el id del responsable de la mitigación no puede ser nulo.	ok

	Error, la fecha límite de la mitigación no puede ser nulo.	ok
	Error, el id de estado no puede ser nulo.	ok
	Error, la short description de mitigación no existe.	ok
BAJA_MITIGACION	Borrado de la mitigación indicada, respetando validaciones.	ok
	Error, la short description de mitigación no existe.	ok
	Error, la short description de mitigación es nula	ok
ALTA_MAESTROPOLITICAS	Alta de una política si no existe, cumpliendo validaciones.	ok
	Error si la short description es nula.	ok
	Error si la política ya está dada de alta.	ok
MODIFICACION_MAESTROPOLITICAS	Modifica las políticas si existen y cumplen las validaciones.	ok
	Error si la short description es nula.	ok
	Error si la política no está dada de alta.	ok
BAJA_MAESTROPOLITICAS	Borra la política indicada en la short description, respetando las validaciones.	ok
	Error si la short description es nula.	ok
	Error si la política no está dada de alta.	ok
ALTA_INCUMPLIMIENTOSPOLITICAS	Alta de un incumplimiento si no existe, cumpliendo validaciones.	ok
	Error si el código de la política es nulo.	ok
	Error si el departamento asociado a la política es nulo.	ok
	Error si el número de incumplimientos de la política es nulo.	ok
	Error si la descripción corta de la política es nula.	ok
	Error si el incumplimiento ya está registrado.	ok
BAJA_INCUMPLIMIENTOSPOLITICAS	Borrado de un incumplimiento si no existe, cumpliendo validaciones.	ok
	Error si el código de la política es nulo.	ok
	Error si la descripción corta de la política es nula.	ok
	Error si el incumplimiento no está registrado.	ok
ALTA_INCUMPLIMIENTOSEMPLEADOS	Alta de los incumplimientos de un empleado, cumpliendo validaciones.	ok
	Error si el id del empleado es nulo.	ok
	Error si la fecha es nula.	ok
	Error si el incumplimiento para esa fecha está ya reportado.	ok

BAJA_INCUMPLIMIENTOSEMPLEADOS	Borrado de los incumplimientos de un empleado en una fecha registrada, cumpliendo validaciones.	ok
	Error si la fecha es nula.	ok
	Error si el incumplimiento para esa fecha está ya reportado.	ok
	Error si el incumplimiento para esa fecha no está ya reportado.	ok
ALTA_DATOS_LOOKUP	Da de alta en bbdd un registro de lookup, cumpliendo validaciones	ok
	Error si la tabla de lookup viene como null..	ok
	Error si el dato a introducir en la tabla dada viene a null.	ok
	Error si la tabla no existe en bbdd.	ok
	Error si el dato de lookup ya está en bbdd, salta restricción única.	ok
CREA_AUDITORIA	Crea una auditoría, cumpliendo validaciones	ok
	Error si quien la realiza viene a null	ok
	Error si la fecha de inicio del periodo a auditar es nula	ok
	Error si la fecha de fin del periodo a auditar es nula	ok
	Error si tipo de auditoría es nulo	ok

2.7.2 Pruebas unitarias respuestas a preguntas planteadas

A continuación generamos un set de pruebas para comprobar que las queries que hemos diseñado para dar respuesta al conjunto de preguntas planteadas arrojan datos correctos. Para ello tomaremos la carga de datos inicial y ampliaremos el set de datos en caso de ser necesario para realizar las comprobaciones oportunas. La forma en la que vamos a comprobar los resultados es haciendo las cuentas con calculadora y ejecutando la query para ver si los resultados obtenidos son los mismos. El fichero con estas pruebas es el llamado “pruebas_unitarias_respuestas_preguntas_planteadas.sql”.

Pregunta #	Resumen de la prueba	Valor esperado	Valor obtenido	ok/ko
1	Con la carga inicia metemos 10 y 10 incumplimientos al id de dpto 2	10 +10 = 20 incumplimientos 2 dpto qué es	El departamento COMPRAS en 2022 tuvo el máximo número	OK

		compras	de incumplimientos... 20	
1	ahora introducimos un año donde no hemos metido incumplimientos	0	No hay departamentos que cumplan dicho criterio en 2021	OK
1	hacemos muchas mas altas de incumplimientos en el 2022 que es el que vamos a probar	55	El departamento CONTROL DE GESTION en 2022 tuvo el máximo número de incumplimientos... 55	OK
2	Con la carga de datos inicial	proceso id 1, tiene 3 vulnerab.	El proceso con id 1 tiene 3 vulnerabilidades.	OK
2	Metemos un id que tenga más vulnerabilidades que las anteriores.	proceso id 2 tiene 4 vulnerab.	El proceso con id 2 tiene 4 vulnerabilidades.	OK
2	Ahora ponemos 2 vulnerabilidades más al proceso id 1 para ver si el bucle con la inicialización funciona.	proceso id 1, tiene 5 vulnerabilidades	El proceso con id 1 tiene 5 vulnerabilidades.	OK
2	Igualamos el proceso 1 y 2 para que tengan el mismo número de vulnerabilidades, ha de mostrarme el primero.	El proceso con id 1 tiene 5 vulnerabilidades.	El proceso con id 1 tiene 5 vulnerabilidades.	OK
2	volvemos a cargar el modelo sin datos para ver cómo se comporta cuando no hay vulnerabilidades	0	No existen procesos con un número máximo de vulnerabilidades.	OK
3	Carga inicial	21Juana, 20 pepe, jose con 20, maria con 20 y raul con 7	El top5 de usuarios por numero de incumplimientos seria... En 1 posicion... JUANA LALOCA con 21 incumplimientos. En 2 posicion... PEPE VIYUELA con 20 incumplimientos. En 3 posicion... JOSE MENDICO con 20 incumplimientos. En 4 posicion... MARIA ARCEIZ con 20 incumplimientos. En 5 posicion... RAUL ALONSO con 7 incumplimientos.	OK
3	Incrementamos en 5 a raul	21Juana, 20 pepe, jose con 20, maria con 20 y raul con 12	El top5 de usuarios por numero de incumplimientos seria... En 1 posicion... JUANA LALOCA con 21 incumplimientos. En 2 posicion... PEPE VIYUELA con 20 incumplimientos. En 3 posicion... JOSE MENDICO con 20 incumplimientos. En 4 posicion... MARIA ARCEIZ con 20 incumplimientos. En 5 posicion... RAUL ALONSO con 12 incumplimientos.	OK

3	Desde la carga inicial añadimos un nuevo incumplimiento al dept 1 de 10. En ese dept. solo está Raul	21Juana, 20 pepe, jose con 20, maria con 20 y raul con17	El top5 de usuarios por numero de incumplimientos seria... En 1 posicion... JUANA LALOCA con 21 incumplimientos. En 2 posicion... PEPE VIYUELA con 20 incumplimientos. En 3 posicion... JOSE MENDICO con 20 incumplimientos. En 4 posicion... MARIA ARCEIZ con 20 incumplimientos. En 5 posicion... RAUL ALONSO con 17 incumplimientos.	OK
4	Con la carga inicial	Tres vulnerabilidades no terminadas.	Hay 3 vulnerabilidades en total. Hay 0 vulnerabilidades acabadas completamente. Un 0% de las vulnerabilidades totalmente mitigadas.	OK
4	Añadimos una acabada	4 en total, de ellas 1 acabada.	Hay 4 vulnerabilidades en total. Hay 1 vulnerabilidades acabadas completamente. Un 25% de las vulnerabilidades totalmente mitigadas.	OK
4	Añadimos otra en otro estado	5 vulneb. solo 1 terminada	Hay 5 vulnerabilidades en total. Hay 1 vulnerabilidades acabadas completamente. Un 20% de las vulnerabilidades totalmente mitigadas.	OK
5	comprobamos cuando se comporta cuando no hay acciones de mitigación, sin cargar datos	0	Hay 0 mitigaciones NO acabadas completamente.	OK
5	con la carga inicial	3	Hay 3 mitigaciones NO acabadas completamente.	OK
5	Añadimos dos mitigaciones más a otra vulnerabilidad	5	Hay 5 mitigaciones NO acabadas completamente.	OK
5	Añadimos 2 mitigaciones más, acabadas	5	Hay 5 mitigaciones NO acabadas completamente.	OK
5	Añadimos 2 mitigaciones más, en proceso	7	Hay 7 mitigaciones NO acabadas completamente.	OK
5	Añadimos 2 mitigaciones más, en revisión	9	Hay 9 mitigaciones NO	OK

			acabadas completamente.	
6	Con la carga inicial	política 1, 15 incumplimientos	La política de seguridad que ha tenido más incumplimientos es la que tiene id. 1 con descripción corta POLITICA1 y tiene 15 incumplimientos.	OK
6	Añadimos 20 incumplimientos a la politica2	política 2, 30 incumplimientos	La politica de seguridad que ha tenido más incumplimientos es la que tiene id. 2 con descripcion corta POLITICA2 y tiene 30 incumplimientos.	OK
7	Con la carga inicial y control de gestión como departamento	1 empleado y no ha terminado la sesión formativa	Hay 1 empleados del departamento Hay 1 empleados del departamento que NO han terminado sesiones formativas El 100% es el porcentaje de los empleados que no han terminado las acciones formativas asignadas para el departamento	OK
7	Con la carga inicial y compras como departamento	4 empleados, todos terminaron la sesión formativa.	Hay 4 empleados del departamento Hay 0 empleados del departamento que NO han terminado sesiones formativas El 0% es el porcentaje de los empleados que no han terminado las acciones formativas asignadas para el departamento	OK
7	Ponemos un departamento que no existe o no tiene empleados	0	No hay empleados en ese dpto. o ese depto no existe	OK
8	Con la carga inicial	5 empleados, 2 con incumplimientos en el año en curso	Hay 5 empleados en total. De ellos, 2 cometen incumplimientos. Por tanto, un 60% no comenten incumplimientos en 2022	OK
8	metemos otro incumplimiento para el año 2021 sobre un empleado	5 empleados, 2 con incumplimientos en el año en curso	Hay 5 empleados en total. De ellos, 2 cometen incumplimientos. Por tanto, un 60% no comenten incumplimientos en	OK

			2022	
8	creamos otro incumplimiento para un empleado existente en 2022	5 empleados, 3 con incumplimientos en el año en curso	Hay 5 empleados en total. De ellos, 3 cometen incumplimientos. Por tanto, un 40% no comenten incumplimientos en 2022	OK
9	con la carga inicial ejecutamos el procedure de generar auditoría	35 en el 2022	El año con mas incumplimientos en todas las auditorias externas registradas fue el: 2022 con 35 incumplimientos.	OK
10	con la carga inicial	1 crítica, sin mitigaciones	Hay 1 vulnerabilidades CRITICAS en total. Hay 0 vulnerabilidades CRITICAS con alguna acción de mitigación abierta. Un 0% de las vulnerabilidades CRITICAS con alguna acción de mitigación abierta.	OK
10	ahora vamos a meter una mitificación no acabada en la crítica	1 crítica con 1 abierta	Hay 1 vulnerabilidades CRITICAS en total. Hay 1 vulnerabilidades CRITICAS con alguna acción de mitigación abierta. Un 100% de las vulnerabilidades CRITICAS con alguna acción de mitigación abierta.	OK
10	metemos una mas a la misma vulnerabilidad, sin terminar	1 crítica con 1 abierta	Hay 1 vulnerabilidades CRITICAS en total. Hay 1 vulnerabilidades CRITICAS con alguna acción de mitigación abierta. Un 100% de las vulnerabilidades CRITICAS con alguna acción de mitigación abierta.	OK
10	metemos una vulnerabilidad mas critica y una acción de mitigación no acabada	2 críticas con 2 abiertas	Hay 2 vulnerabilidades CRITICAS en total. Hay 2 vulnerabilidades CRITICAS con alguna acción de mitigación abierta. Un 100% de las vulnerabilidades CRITICAS con alguna acción de mitigación	OK

			abierta.	
11	carga de datos inicial con 0 sesiones	0 sesiones telemáticas	0 sesiones	OK
11	Añadimos nuevas sesiones telemáticas	ha de salir 4 sesiones y la 5 es la que menos empleados asisten	Hay 4 inscritos a las diferentes sesiones de formacion en total. La sesion con menor porcentaje de participantes es... TITULO5 SESIONFORMATIVA con 1 participantes. TITULO5 SESIONFORMATIVA tiene un porcentaje de participacion del 25% de los inscritos a alguna sesion formativa.	OK
12	Carga Inicial	1	1 Vulnerabilidades CRITICAS fueron encontradas internamente despues del analisis inicial realizado por la consultora externa, despues del 05/01/22	OK
12	revisamos si ok con nulo	0	0 Vulnerabilidades CRITICAS fueron encontradas internamente despues del analisis inicial realizado por la consultora externa, despues del 24/06/22	OK
13	Carga inicial	0 con estados 2 o 4	Hay 3 mitigaciones en total. Hay 0 mitigaciones en estados "EN PROCESO" o "EN REVISION". Lo cual supone que hay un 0% del total de mitigaciones en estado "EN PROCESO" o "EN REVISION".	OK
13	Damos de alta una de estado en proceso y otra en revisión	nos tiene que dar 2	Hay 5 mitigaciones en total. Hay 2 mitigaciones en estados "EN PROCESO" o "EN REVISION". Lo cual supone que hay un 40% del total de mitigaciones en estado "EN PROCESO" o "EN REVISION".	OK
14	Añadimos a otro usuario mas acciones de mitigación	Raul Alonso es el que más tiene	El empleado con Id. 1 RAUL ALONSO tiene 1	OK

			acciones de remediacion "en proceso", es el que mas tiene.	
14	Carga inicial	0	No hay empleados que cumplan los criterios	OK

3. Conclusiones

3.1 Seguimiento del proyecto.

- **PEC1**.- La entrega se realizó sin problema, la extensión en días de la PEC fué suficiente para que de una forma desahogada cumplieramos los tiempos establecidos. El profesor echó en falta la elaboración de un plan de contingencia y las acciones que mitigarán/eliminarán los posibles problemas que nos podríamos encontrar durante el desarrollo del proyecto. También se transmitió la duda sobre la idoneidad de la metodología seleccionada para el desarrollo del TFG. El primer punto fué desarrollado en la entrega posterior que tuvimos que realizar y el segundo conseguimos adecuar la metodología scrum a un proyecto tan corto como el nuestro de una forma imaginativa y flexible.
- **PEC2**.- El planning establecido se ha cumplido y se ha entregado más material del inicialmente planificado, gracias en gran medida, a la aplicación de scrum en el desarrollo del proyecto. El profesor nos recomienda en su valoración un nuevo apartado que sería el seguimiento de la planificación. He aquí, en estas líneas, el mismo. Respecto a la necesidad de tener en mente las consultas que se nos piden en el enunciado a la hora de hacer el modelo de datos, lo hemos hecho, de ahí las preguntas en el foro sobre si se podían usar los count(*) p.ej, muchas gracias por incidir en este detalle que tiene gran importancia y que nos ahorrará tener que remodelar en algún caso.
- **PEC3**.- Se han cumplido los hitos propuestos en el tiempo marcado. Se ha realizado sin excesiva complejidad el volcado del modelo físico a la bbdd Oracle, se ha creado un mini set de datos para probar los procesos almacenados y poder trabajar sobre las consultas. Una vez comprobada la funcionalidad de las consultas para ese set de datos manejable, se incrementará en la entrega final el número de registros viendo la efectividad de las mismas también con más registros. Entregamos también en esta entrega casi la totalidad de las preguntas que se nos han formulado en el enunciado, todo encuadrado dentro del planning acordado inicialmente. Queda pendiente para finalizar la memoria las pruebas unitarias de los paquetes creados, resolver el par de queries pendientes y adjuntar las conclusiones del TFG.
- **Entrega final**.- Como siempre, hemos valorado muy positivamente el feedback recibido por nuestro profesor y hemos procedido a contar las palabras del resumen del trabajo comprobando que no superamos las

250. Por otro lado, hemos procedido a sacar de la memoria el proceso de instalación del entorno a un fichero a parte, tal como se nos sugería. Sin embargo no hemos estado muy de acuerdo en poner la bibliografía a pie de página ya que esto solemos hacerlo con citas, que en este trabajo son inexistentes. Por tanto, se ha mantenido la bibliografía al final del documento. Del desarrollo de estos últimos sprints, lo único que podríamos destacar es que hemos de adelantar el trabajo que teníamos pensado realizar en la semana del 6-10 de junio ya que por viaje de negocios no podremos dedicarle tiempo. Hemos aplicado la medida correctiva que definimos en el inicio de la memoria que es tomar vacaciones los días 23 y 24 de mayo con el fin de disponer en esos días del tiempo que no vamos a tener en Junio. Adicionalmente hemos de decir que en la planificación inicial no tuvimos en cuenta la necesidad de la creación de la presentación del TFG, por lo que en estos días de vacaciones el los que hemos de realizar las tareas que no vamos a poder hacer por el viaje, tendremos que sacar 5 ó 6 horas para la creación del powerpoint.

3.2 Conclusiones del proyecto.

Hemos terminado por fin, un largo camino sin duda desde que empezamos el Grado. Con la finalización de este TFG damos por concluida la etapa formativa que empezó hace ya unos años.
Este TFG ha sido muy enriquecedor ya que ha permitido poner en marcha y en práctica todo lo aprendido en la carrera, desde la estimación de tiempos, pasando por la toma de requisitos, continuando por el diseño y finalizando con la implantación y pruebas.
Del proyecto hemos podido aprender que también somos capaces, de una manera individual, de crear un producto informático desde cero. Muchas veces en la vida laboral que tenemos sólo eres parte de un engranaje y nunca alcanzas a ver todo el conjunto, este TFG nos ha permitido, durante su realización, ser el analista funcional, analista programador, programador, jefe de proyecto y chico de sistemas… todo a la vez y en la misma persona.

Creo que si hemos conseguido alcanzar los objetivos planteados en el enunciado del TFG. El producto final es funcional, quizás nos hubiera gustado disponer de más tiempo para poder añadir más funcionalidades de las requeridas por el proyecto y profundizar más, por ejemplo, en la parte de las auditorías que en el enunciado no se desarrolla profundamente. Ha sido una pequeña locura el no haber podido usar funciones de agregación, que sin duda, hubieran acortado los tiempos necesarios para el desarrollo. Pero a cambio hemos tenido que desplegar una gran imaginación con el uso de contadores, tablas temporales, etc. que han hecho mucho más interesante esta parte.

La planificación se ha cumplido de una manera eficaz y la metodología agile “personalizada” que inicialmente planteaba ciertas dudas para un desarrollo tan pequeño… ha funcionado muy bien. Hemos conseguido a partir de la

segunda entrega proporcionar código funcional a nuestro profesor, que nos ha ido proporcionando feedback para ir mejorando el producto. Si hubiera usado waterfall es fácil que hasta la tercera entrega no hubiéramos obtenido feedback del código.

Quizás hemos echado de menos, y podría ir en las líneas de trabajo futuro, no haber tenido que desarrollar una pequeña aplicación gráfica que usara nuestro modelo de datos. Tal vez un APEX enganchado a él, que de una forma muy sencilla nos permite crear pantallas atractivas y más vistoso nuestro proyecto.
También nos hubiera gustado crear un mayor número de funciones de validación, como por ejemplo implementar la función de validación del dni, cuyo código si que he creado, a la hora de insertar el NIF.

4. Glosario

Trigger.- Es una serie de reglas predefinidas que se asocian a una tabla. Estas reglas se aplican a la base de datos cuando se realizan determinadas operaciones en la tabla, por ejemplo, al añadir, actualizar o eliminar registros.
UML.- UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).
Foreign Key.- Una FOREIGN KEY es una clave (campo de una columna) que sirve para relacionar dos tablas. El campo FOREIGN KEY se relaciona o vincula con la PRIMARY KEY de otra tabla de la bbdd.
Primary Key.- En el diseño de bases de datos relacionales, se llama clave primaria o llave primaria o clave principal a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria.
Procedimiento almacenado.- Programa almacenado en base de datos que se ejecuta directamente en el motor de bases de datos.
SQL.- Es un acrónimo en inglés para Structured Query Language. Un Lenguaje de Consulta Estructurado. Un tipo de lenguaje de programación que permite manipular y descargar datos de una base de datos.
Diagrama de Gantt.- Un diagrama de Gantt es una herramienta útil para planificar proyectos. Al proporcionar una vista general de las tareas programadas, todas las partes implicadas sabrán qué tareas tienen que completarse y en qué fecha.
Secuencia.- Su objetivo principal es generar el valor de clave principal de la tabla, al que se puede hacer referencia en la instrucción de inserción, también puede verificar el valor actual a través de una consulta o aumentar la secuencia al siguiente valor.

5. Bibliografía

- Material de Bases de Datos I
  Jordi Conesa Caralt, Angels Rius Gavidia, M. Elena Rodriguez González
  UOC 1ª Edición (febrero 2011)
- Material de Bases de Datos II
  Angels Rius Gavidia (coordinadora)
  UOC 4ª Edición (septiembre 2015)
- Material de Ingeniería del Software I
  Jordi Pradel Mikel, José Raya Martos
  UOC 2ª Edición (septiembre 2014)
- Material de Gestión de Proyectos
  Jose Ramón Rodriguez (coordinador) , Pere Mariné Jové
  UOC 2ª Edición (septiembre 2014)
- Oracle Database Online Documentation 21c. [En línea]
  https://docs.oracle.com/en/database/oracle/oracle-database/
- Guia de Usuario de SQL Developer [En línea]
  https://docs.oracle.com/cd/E12151_01/doc.150/e12152.pdf
- Modelo lógico:
  https://virtual.itca.edu.sv/Mediadores/dbd/u1/14_mdelo_lgico_de_datos.html