colegio CALASANZ SALAMANCA

Paseo de Canalejas, 139-159
37001 Salamanca
Tel.: 923 26 79 61 | Fax: 923 27 04 54
www.calasanzsalamanca.es
CENTRO PRIVADO CONCERTADO

RECOMENDACIONES Y CRITERIOS GENERALES DE CORRECCIÓN:

- El proyecto y base de datos tendrán el formato: **PracticaFinal+[1ª apellido 1º] +[1ª apellido 2º] +[1ª y 2ª nombre o compuesto]**
    - Ejemplo: Ibam
- El entregable debe **compilar**, si no es así supondrá un 0 automático.
- El **plagio** supondrá un 0 tanto para el plagiador como para el plagiado
- Se deben respetar las **reglas de estilo** (por cada regla que no se respete restará 0,5 puntos):
    - Los nombres de las **clases** deben ser nombres y estar en PascalCase y estar en ficheros separados.
    - Los nombres de los **métodos** tienen que ser verbos y estar el camelCase.
    - Los nombres de las **variables** tienen que ser descriptivas del valor que guardan y estar en camelCase.
    - Se debe respetar los niveles de sangrado
- Se recomienda el uso del *this*

Enunciado: Aplicación de Control de Gastos de Coches

Bueno, le explico lo que necesitamos para esta aplicación que vamos a usar para llevar el control de los gastos de nuestros coches. Mire, la cosa es sencilla pero tiene que estar bien hecha, ¿eh?

Lo primero que tenemos son los usuarios. La gente se tiene que poder registrar en el sistema con su nombre, que tiene que ser único, claro está. No puede haber dos personas con el mismo nombre en el sistema. Además, cada usuario cuando se registre, el sistema le tiene que asignar automáticamente un código UUID, que es como su identificación única. Este código es importante porque lo van a necesitar después para otras cosas.

Una vez registrados, los usuarios se tienen que poder conectar al sistema, o sea, hacer login con sus datos. Nada complicado, pero tiene que funcionar bien y ser seguro.

Ahora viene lo del tema de los coches. Cada usuario puede crear coches en el sistema, ¿vale? Cuando crea un coche, automáticamente se convierte en propietario de ese coche. Pero aquí está la gracia: un coche puede tener varios propietarios. ¿Cómo se hace esto? Pues el propietario original puede añadir a otros usuarios como propietarios también, pero para eso necesita conocer el código UUID de esa otra persona. No le vamos a mostrar una lista de todos los usuarios ni nada de eso, tiene que saber el código de la persona que quiere añadir.

Los coches, pues tendrán la información típica: marca, modelo, matrícula, año... lo normal que uno esperaría encontrar sobre un coche. Y cada coche puede tener esos múltiples propietarios que le decía.

Luego está el tema de los gastos, que es lo importante. Los usuarios pueden ver todos los coches de los que son propietarios, los que han creado ellos o en los que les han añadido. Para cada uno de estos coches, pueden hacer varias cosas: ver la información, editarla si hace falta, eliminar el coche si es necesario, y lo más importante, añadir gastos.

Los gastos tienen que tener cierta información obligatoria. Primero, el tipo de gasto, pero no vale poner cualquier cosa. Tienen que elegir de una lista que ya está definida: echar gasolina, revisión, ITV, cambio de aceite u otros. Solo esas opciones, nada más. También hay que poner el kilometraje que tenía el coche en ese momento del gasto, la fecha de cuando se hizo el gasto, y el importe, claro. Opcionalmente se puede añadir una descripción por si quieren poner algún detalle extra.

Ah, y por cada coche que tenga en propiedad, el usuario tiene que poder ver un resumen de todos los gastos. Esto se presenta en forma de tabla, sin gráficos ni nada fancy, solo una tabla con los datos. Pero aquí viene lo bueno: se pueden poner filtros. Como mínimo tienen que poder filtrar por año y por fecha, y también por los kilómetros. Los filtros pueden estar en "Todos" si no quieren filtrar nada específico.

Esta vista de gastos es importante porque le permite al usuario ver cómo va gastando en cada coche, cuándo fue la última revisión, cuánto se gastó en gasolina el año pasado, ese tipo de cosas útiles.

El sistema tiene que controlar bien los errores. Si alguien intenta hacer algo que no puede, o si se equivoca metiendo datos, o si hay algún problema con la base de datos, el sistema tiene que responder de manera adecuada y no cascarse.

Y bueno, todo esto se tiene que hacer siguiendo ese patrón MVC que hemos estado viendo en clase, ¿eh? La vista puede ser lo que les resulte más cómodo: por consola, con Swing o JSP, como prefieran. Los datos se guardan en MySQL usando JDBC, como hemos hecho siempre.

La aplicación tiene que funcionar de forma que varios usuarios puedan usar el sistema, cada uno con sus coches y sus gastos, pero también permitir esa colaboración cuando un coche tiene varios propietarios. Todos los propietarios de un coche pueden ver y gestionar los gastos de ese coche.

Resumiendo, necesitamos gestionar usuarios, coches, la relación entre usuarios y coches (porque puede ser de muchos a muchos), y los gastos de cada coche. Todo tiene que estar bien conectado y funcionando correctamente, con buena organización del código y control de errores.

Entrega

El proyecto se entregará a través de Git siguiendo la estructura de carpetas y organización que hemos utilizado durante el curso.

Debe incluir:

- Un archivo README con las instrucciones claras para ejecutar la aplicación (creación de base de datos, importación del proyecto, ejecución...)
- Los archivos SQL necesarios para crear la base de datos e insertar los datos iniciales que sean necesarios
    - No es necesario incluir diagramas de base de datos, con los scripts SQL es suficiente.
- Todo el código fuente organizado según el patrón MVC establecido en clase