Chuletas para el MQ

Projecte d'Enginyeria del Software (Universitat Politècnica de Catalunya)

Ejercicios de PES

1. Define Play Java framework.

Play java framework es una aplicación web de código abierto que sigue el patrón model-view-controller.

2. Explica que son los patrones de diseño y pon un ejemplo.

Un patrón de diseño es una solución general a un problema común i recurrente en el diseño de software.

Un ejemplo de patrón de diseño es el proceso de Log In.

3. Que es Hibernate y para que se utiliza en Play java Framework.

Hibernate es una solución implementada para el mapeo objeto-relacional (ORM) sobre una base de datos relacional.

Hibernate se usa en Play java framework para mapear las clases de Java a tablas de la base de Datos.

10. Una peti
e inconvenie

Les respostes correctes són: Hibernate és una implementació de JPA, Hibernate és una eina ORM basada en Java que proporciona un marc per al mapeig d'objectes java a taules de bases de dades relacionals i viceversa.

La petición h
cuerpo donde se encuentra el fichero html al que va dirigido esa peticion.

El método GET se usa para obtener un recurso específico e incluye toda la información en la cabecera del mensaje.

Este método se podría usar en la acción de un usuario para obtener información acerca de un producto.

El método POST se usa para crear un recurso nuevo e incluye solo la información del servidor y de la cabecera del mensaje.

Este método se podría usar en la acción de crear un nuevo evento.

Les cookies HTTP són petits fragments de dades que el servidor web envia al navegador web de l'usuari i que es poden emmagatzemar localment al dispositiu de l'usuari. Quan l'usuari torna a visitar el mateix lloc web, el navegador envia les cookies associades amb aquest lloc web de nou al servidor. Les cookies són utilitzades per a una àmplia varietat de finalitats, com ara mantenir l'estat de la sessió de l'usuari (com l'informació d'inici de sessió), personalitzar la experiència de l'usuari, recopilar dades d'ús del lloc web i per altres funcionalitats.

En resum, mentre que el protocol HTTP és "sense estat" en el sentit que cada petició és tractada independentment, les cookies HTTP permeten als llocs web mantenir i gestionar l'estat de les sessions dels usuaris i proporcionar una experiència més personalitzada i funcional als usuaris.

·*Patrón de diseño MVC Modelo Vista Controlador*

Es un patrón de diseño de software que separa: los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Modelo: Contiene los datos de la aplicación.
Vista: Presenta los datos almacenados en el modelo.
Controlador:

1.- Implementa la lógica para procesar/aceptar la entrada del usuario.
2.- Define como reacciona la interfaz de usuarios a dicha entrada.
3.- Selecciona la próxima View para el usuario.

Relación->clau forana
clase modelo -> clau primaria
clase -> taula

5. La anotacion @Entity que efectos tiene en el entorno de programación Play java framework?

Cuando se crea una clase la anotación @Entity la transforma en una tabla de la base de datos relacional de forma automática.

Que hauríem d'afegir per a que el nombre de taules de la base de dades relacional creada fós òptim (mínim nombre taules i sense valors duplicats)

Resposta: mappedBy ✔

Como sería la petición http que realiza una petición al servicio ListaMedicosHospital? Qué método http se utiliza cuando se realiza una petición http des de la ventana de búsqueda del navegador?

localhost:9000/Application/ListaMedicosHospital?nomH=Vall d'Hebron

GET

La resposta correcta és: JPA proporciona una especificació per persistir, llegir i gestionar dades d'objectes Java a taules relacionals de la base de dades.

2. **Quins problemes veus en aquest disseny? Com ho solucionaries**

Tenim la següent informació d'una base de dades:

ID	NOM_CLIENT	DNI	EDAT
1	lola	8989	54
2	pedro	8899	18

ID	NOM_PRODUCTE	PREU	COMPRADOR
3	producte1	200	lola
4	producte2	25	Pedro

A la segona taula apareix el nom del comprador, quan aquest camp no faria falta si podem crear i definir una relació entre les taules.

```
@Entity
public class Producte extends Model{
public string nom_Producte;
public int preu;
@OneToOne
public client comprador;
public Producte(string n, int p){
nom_Producte = n;
preu = p;
}
```

15. **Qué és JPA (Java Persistence Api) ?**

JPA proporciona una especificació per persistir, llegir i gestionar dades d'objectes Java a taules relacionals de la base de dades.
JPA és un entorn de treball per gestionar Bases de dades.
JPA és un llenguatge que ens permet gestionar la Base de dades de un programa.

7. **A quin llenguatge pertanyen les següents anotacions?**
Per a què has utilitzat les següents anotacions en el teu projecte? i si no ho has fet a on les utilitzaries i per a què?
@Entity
@ManyToMany, @OneToMany, @OneToOne
@Before
@OnApplicationStart

Pertanyen al llenguatge Hibernate.
Les hem utilitzat per:
@Entity: Indicat que una classe s'ha de transformar en BBDD.
@ManyToMany, @OneToMany, @OneToOne: Definir la relació entre les taules, ja sigui N:N, 1:N o 1:1
@Before:
@OnApplicationStart:

2. **Quins problemes veus en aquest disseny? Com ho solucionaries**

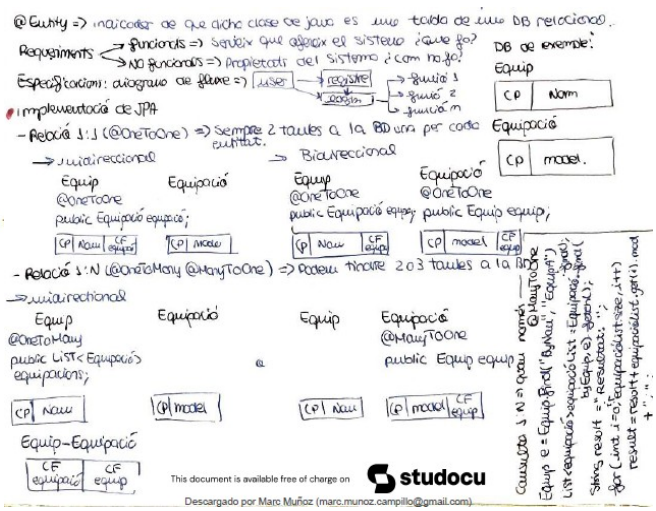
Tenim la següent informació d'una base de dades:

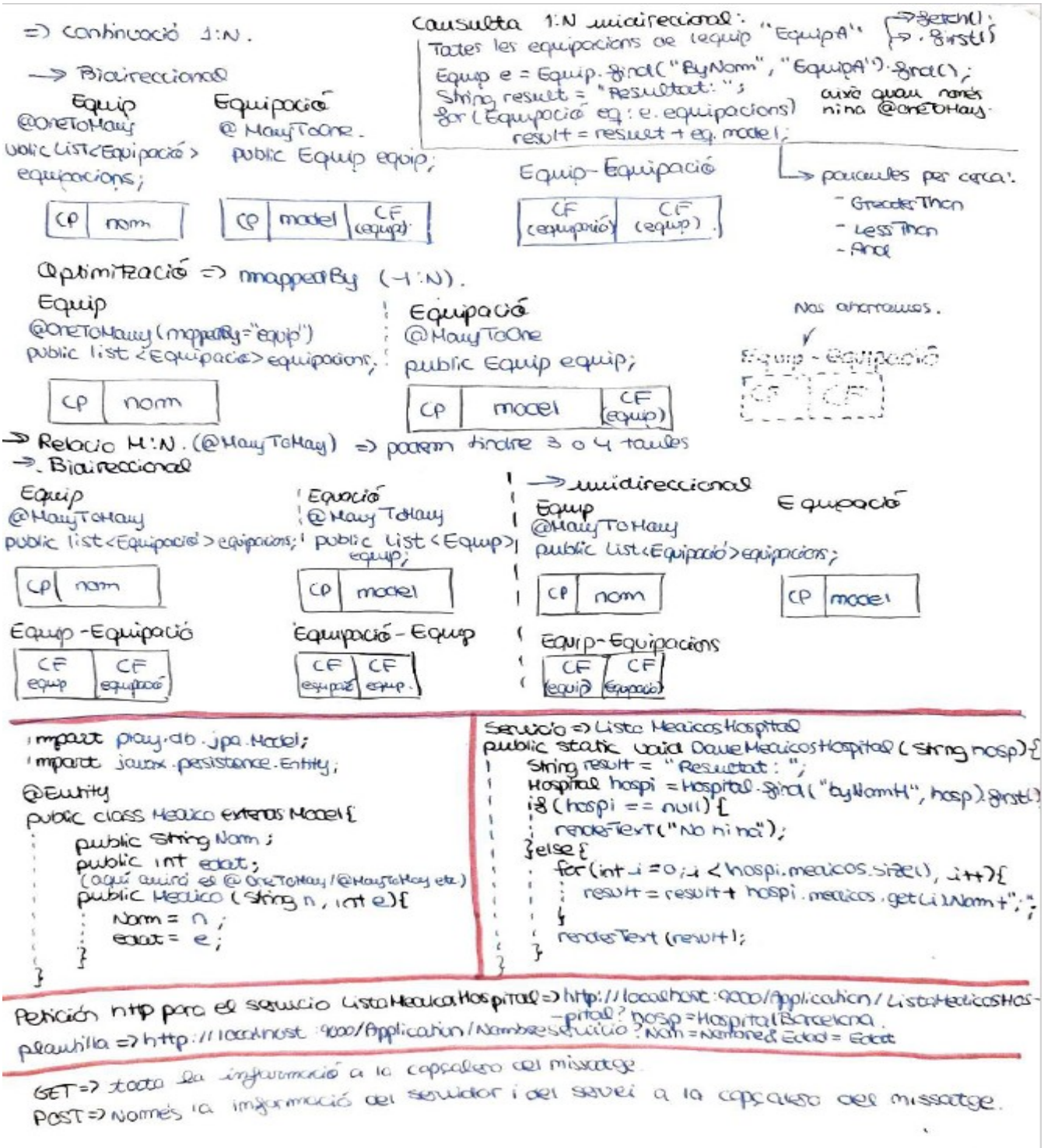
ID	NOM_CLIENT	DNI	EDAT
1	lola	8989	54
2	pedro	8899	18

ID	NOM_PRODUCTE	PREU	COMPRADOR
3	producte1	200	lola
4	producte2	25	Pedro

A la segona taula apareix el nom del comprador, quan aquest camp no faria
podem crear i definir una relació entre les taules.

```
@Entity
public class Producte extends Model{
public string nom_Producte;
public int preu;
@OneToOne
public client comprador;
public Producte(string n, int p){
nom_Producte = n;
preu = p;
}
```






```
import play.db.jpa.Model;
import javax.persistence.Entity;

@Entity
public class Medico extends Model{
    public String Nom;
    public int edat;
    (aquí aniria es @OneToMany/@ManyToMany etc.)
    public Medico (String n, int e){
        Nom = n;
        edat = e;
    }
}
```

```
Servicio => ListaMedicosHospital
public static void DameMedicosHospital(String nosp){
    String result = "Resultat: ";
    Hospital hospi = Hospital.find("byNomH", hosp).first();
    if(hospi == null){
        renderText("No hi ha");
    }else{
        for(int i=0; i<hospi.medicos.size(); i++){
            result = result + hospi.medicos.get(i).Nom+";";
        }
        renderText(result);
    }
}
```

Petición http para el servicio ListaMedicosHospital => http://localhost:9000/Application/ListaMedicosHospital?nosp=HospitalBarcelona.

plantilla => http://localhost:9000/Application/NombreServicio?Nom=NombreA&Edad=Edad

GET => toda la información a la capçalera del missatge.

POST => Només la informació del servidor i del servei a la capçalera del missatge.

Hibernate: Relacions Unidireccionals i Bidireccionals
PES - Daniel García
13/03/2021

1. OneToMany

Realitza les següents activitats.

1. Descomprimeix el fitxer OneToMany.zip en el directori Play-1.5.3.
2. El nou servidor que apareix té per nom OneToMany i com que ha estat creat en un entorn diferent del vostre heu d'executar el següent comanda: prompt> play dependencies OneToMany
3. Obriu el projecte anb el IDE que tingueu instal.lat (Intellij, Eclipse) Veureu que el model de dades conté dues classes/entitats: Propietari i Cotxe.

Considerem que la relació entre les dues entitats és 1:N. Un propietari pot tenir molts cotxes però un cotxe tan sols pot tenir un propietari.

1.1. Relació Unidireccional en la entitat Cotxe.

```
<Cotxe>
@ManyToOne
public Propietari propietari;
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen dues taules: Cotxe (amb propietari_ID) i Propietari.

`Cotxe c = new Cotxe("Peugeot", "6052KKL");` `Propietari p = new Propietari("Carlos Manuel",18).save();`	//Fem dues noves instancies de les clases definides i les guardem.
`c.propietari = p;` `c.save();`	//La instancia c de la clase Cotxe té una relació anomenada propietari que el relacionem amb el propietari de la instancia p de la clase Propietari i guardem els canvis.

1.2. Relació Unidireccional en la entitat Propietari

```
<Propietari>
@OneToMany
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es crean tres taules: Cotxe, Propietari, Propietari_Cotxe.

`Cotxe c = new Cotxe("Peugeot", "6052KKL").save();` `Propietari p = new Propietari("Carlos Manuel",18);`	//Fem dues noves instancies de les clases definides i les guardem.
`p.cotxeList.add(c);` `p.save();`	//A la llista cotxeList creada a la clase de la instancia p li afegin la instancia de la clase cotxe c i guardem els canvis.

1.3. Relació Bidireccional sense indicar qui és el propietari de la relació

```
<Cotxe>
@ManyToOne
public Propietari propietari;

<Propietari>
@OneToMany
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es crean tres taules: Cotxe (amb propietari_ID), Propietari, Propietari_Cotxe.

`Cotxe c = new Cotxe("Peugeot", "6052KKL").save();` `Propietari p = new Propietari("Carlos Manuel",18).save();`	//Fem dues noves instancies de les clases definides i guardem.
`c.propietari = p;` `c.save();`	//Asigmen la relació com a la qüesió 1.1. i guardem.
`p.cotxeList.add(c);` `p.save();`	//A la llista cotxeList creada a la clase de la instancia p li afegin la instancia de la clase cotxe c i guardem.

1.4. Relació Bidireccional indicant qui és el propietari de la relació

```
<Cotxe>
@ManyToOne
public Propietari propietari;

<Propietari>
@OneToMany (mappedBy = "propietari")
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

//Amb aquest mappedBy li otorgo la propietat de la relació a la relació propietari creada a la clase Cotxe.

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen dues taules: Cotxe (amb propietari_ID) i Propietari. Es l'opció més completa i eficient.

Exactament igual a la qüestió 1.3.

2. ManyToMany (Una persona pot tenir varis cotxes y un cotxe pot tenir varis propietaris)

2.1. Relació Unidireccional en la entitat Cotxe.

```
<Cotxe>
@ManyToMany
public List<Propietari> propietariList = new ArrayList<Propietari>();
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen tres taules: Cotxe, Cotxe_Propietari i Propietari.

`Cotxe c = new Cotxe("Peugeot", "6052KKL");` `Propietari p = new Propietari("Carlos Manuel",18).save();`	//Fem dues noves instancies de les clases definides i guardem..
`c.propietariList.add(p);` `c.save();`	//Afegim a la llista de propietaris creada a la clase Cotxe la instancia de Propietari p relacionada amb la instancia de Cotxe c i guardem els canvis.

2.2. Relació Unidireccional en la entitat Propietari

```
<Propietari>
@ManyToMany
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen tres taules: Cotxe, Propietari i Propietari_Cotxe.

`Cotxe c = new Cotxe("Peugeot", "6052KKL").save();` `Propietari p = new Propietari("Carlos Manuel",18);`	//Fem dues noves instancies de les clases definides i guardem.
`p.cotxeList.add(c);` `p.save();`	//Afegim a la llista de cotxes creada a la clase Propietari la instancia de Cotxe c relacionada amb la instancia de Propietari p i guardem.

2.3. Relació Bidireccional sense indicar qui és el propietari de la relació

```
<Cotxe>
@ManyToMany
public List<Propietari> propietariList = new ArrayList<Propietari>();

<Propietari>
@ManyToMany
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen quatre taules: Cotxe, Cotxe_Propietari, Propietari i Propietari_Cotxe.

`Cotxe c = new Cotxe("Peugeot", "6052KKL").save();` `Propietari p = new Propietari("Carlos Manuel",18).save();`	//Fem dues noves instancies de les clases definides i les guardem.

`c.propietariList.add(p);`
`p.cotxeList.add(c);`
`c.save();`
`p.save();`
	//Afegim a les llistes i guardem els canvis..

2.4. Relació Bidireccional indicant qui és el propietari de la relació

```
<Cotxe>
@ManyToMany
public List<Propietari> propietariList = new ArrayList<Propietari>();

<Propietari>
@ManyToMany (mappedBy = "propietariList")
public List<Cotxe> cotxeList = new ArrayList<Cotxe>();
```

//Amb aquest mappedBy li otorgo la propietat de la relació a la llista propietari creada a la clase Cotxe, en aquest cas sería indeferent posar el mappedBy a l'altre relació.

Quantes taules es creen a la base de dades? Com s'afegeix un cotxe i un propietari a la base de dades?

Es creen tres taules: Cotxe, Cotxe_Propietari i Propietari.