Classe PezzoEntity

```
@Entity(name="Pezzo")
@NamedQueries({
    @NamedQuery(name=TROVA_TUTTI, query="SELECT p FROM Pezzo p"),
    @NamedQuery(name=TROVA_PER_ID, query="SELECT p "
        + "FROM Pezzo p WHERE p.id = ?1"),
    @NamedQuery(name=TROVA_PER_CATEGORIA, query="SELECT p"
        + "FROM Pezzo p WHERE p.categorie = :categorie")
})
@XmlRootElement
public class PezzoEntity implements Serializable {
    public static final String TROVA_TUTTI="Pezzo.TrovaTutti";
    public static final String TROVA_PER_ID="Pezzo.TrovaPerId";
    public static final String TROVA_PER_CATEGORIA="Pezzo.TrovaPerCategorie";
    // COSTRUTTORE (PIENO E VUOTO)
    // GETTER E SETTER
    // TO STRING
    @Id @GeneratedValue
    private int id;
```

Scansionato con CamScanner

```
@NotNull
@Column (name = "nome", nullable = false)
private String nomeRicambio;
@NotNull @Enumerated (EnumType.STRING)
@Column (nullable = false)
private Categoria categoria;
@NotNull @Min (0)
@Column (nullable = false)
private double prezzo;
@NotNull @Min (0)
@Column (nullable = false)
private int disponibilità;
@NotNull @Min (0)
@Column (nullable = false, name = "rendite")
private int pezziVenduti;
public enum Categoria {
        MOTORE, FRENI, FILTRI, CARROZZERIA, ACCENSIONE,
        SOSPENSIONI, IMPIANTO ELETTRICO }
}
```

Scansionato con CamScanner

Interfaccia PrezzoEJBRemote

```
@Remote
public interface PrezzoEJBRemote {
    void creaPrezzo(PrezzoEntity p);
    PrezzoEntity aggiornaPrezzo(PrezzoEntity p);
    void rimuoviPrezzo(PrezzoEntity p);
    PrezzoEntity ottieniDaId(int id);
    List<PrezzoEntity> ottieniTutti();
    List<PrezzoEntity> ottieniDaCategorie(Categoria categorie);
    List<PrezzoEntity> ottieniDaComprare();
    PrezzoEntity aggiornaPrezzo(int id, double prezzo);
}
```

Interfaccia Counter

```
@InterceptorBinding
@Target({METHOD, TYPE})
@Retention(RUNTIME)
public @interface Counter { }
```

Scansionato con CamScanner

Classe PezzoESB

```
@Stateless
@LocalBean
@Remote
@WebService
public class PezzoESB implements PezzoESBRemote {
    @Inject
    private EntityManager em;
    @Override
    public void creaPezzo (PezzoEntity p){
        em.persist(p);
    }
    @Override
    public PezzoEntity aggiornaPezzo (PezzoEntity p){
        return em.merge(p);
    }
    @Override
    public void rimuoviPezzo(PezzoEntity p) {
        em.remove(em.merge(p));
    }
```

Scansionato con CamScanner

```
@Override
public PezzoEntity getById(int id){
    return em.createNamedQuery(
        PezzoEntity.TROVA_PER_ID, PezzoEntity.class)
        .setParameter(1, id)
        .getSingleResult();
}
@Override
public List<PezzoEntity> ottieniTutti(){
    return em.createNamedQuery(
        PezzoEntity.TROVA_TUTTI, PezzoEntity.class)
        .getResultList();
}
@Override
public List<PezzoEntity> ottieniDaCategoria(Categoria categoria){
    return em.createNamedQuery(
        PezzoEntity.TROVA_PER_CATEGORIA, PezzoEntity.class)
        .setParameter("categoria", categoria)
        .getResultList();
}
```

Scansionato con CamScanner

```
@Override
public List<PrezzoEntity> ottieniDaComprare() {
    return em.createQuery(
        "SELECT p FROM Prezzo p WHERE p.disponibilità < 10",
        PrezzoEntity.class)
        .getResultList();
}
@Override
@WebMethod(operationName="updatePrice")
@WebResult(name="itemUpdated")
public PrezzoEntity aggiornaPrezzo(int id,
    @WebParam(name="price") double price) {
    PrezzoEntity p = ottieniDaId(id);
    p.setPrezzo(prezzo);
    return p;
}
}
```

Scansionato con CamScanner

Classe DatabaseProducer

```
public class DatabaseProducer {
    @Produces
    @PersistenceContext (unitName = "EsamePU")
    private EntityManager em;
}
```

Classe DatabasePopulator

```
@Singleton
@Startup
@DataSourceDefinition (
    className = "org.apache.derby.jdbc.EmbeddedDataSource",
    name = "java:global/jdbc/EsameDS",
    user = "app", password = "app",
    databaseName = "EsameDB",
    properties = {"connectionAttributes=;create=true"}
)

public class DatabasePopulator {
    @Inject
    private PersoEJB pjb;
```

Scansionato con CamScanner

```
private PezzoEntity p1, p2, p3;
@PostConstruct
private void populateDB(){
    //INIZIALIZZA p1, p2, p3
    ejb.creaPezzo(p1);
    ejb.creaPezzo(p2);
    ejb.creaPezzo(p3);
}
@PreDestroy
private void clearDB(){
    ejb.rimuoviPezzo(p1);
    ejb.rimuoviPezzo(p2);
    ejb.rimuoviPezzo(p3);
}
}
```

Scansionato con CamScanner

Classe CounterInterceptor

```
@Interceptor
@Counter
public class CounterInterceptor {
    private int counter = 0;
    @AroundInvoke
    public Object logMethod(InvocationContext ic) throws Exception {
        ++counter;
        System.out.println("Chiamata n. " + counter + " : metodo "
            + ic.getMethod().getName());
    }
}
```

Scansionato con CamScanner

Classe PezzoMDB

```
@MessageDriven (mappedName = "jms/queueFiTque")
public class PezzoMDB implements MessageListener {
    @Inject
    private PezzoEJB ejb;
    @Inject
    private Event<PezzoEntity> e;
    @Override
    public void onMessage (Message message) {
        try {
            int id = message.getIntProperty ("id");
            int vendite = message.getBody (Integer.class);
            PezzoEntity p = ejb.ottieniDaId(id);
            p.setPezziVenduti (p.getPezziVenduti() + vendite);
            e.fire(p);
        } catch (JMSException e) {
            // GESTIRE ECCEZIONE
        }
    }
}
```

Scansionato con CamScanner

Classe UpdateNotification

```
public class UpdateNotification {
    public void notify(@Observes PezzoEntity p){
        System.out.println("Le vendite del pezzo " + p.getId()
            + " è stato aggiornato. Vendite: " + p.getPezziVenduti());
    }
}
```

STRUTTURA PROGETTO SERVER, NOME PROGETTO: GRILLOANTONIO_EJB

- SRC
  - it.unisa
    - PEZZOENTITY
    - PEZZOESB
    - PEZZOESBREMOTE
    - DATABASEPRODUCER
    - DATABASEPOPULATOR
    - PEZZOMDB
    - COUNTER
    - COUNTERINTERCEPTOR
    - UPDATENOTIFICATION
- LIBRARY
  - JAVA EE 8 API
- CONF
  - BEANS.XML
    - PERSISTENCE.XML

Scansionato con CamScanner

Nel file beans.xml sostituiamo "ANNOTATED" con "ALL", e aggiungiamo il modo <interceptors>

<class> it.unisa.CounterInterceptor </class>

</interceptors>

Nel file persistence.xml dichiariamo la persistence unit con il nome "Esame"

aggiungiamo il JNDI del DataSource:

<jta-data-source>

java:global/jdbc/EsameDS

</jta-data-source>.

Specifichiamo il target del database uguale a DERBY.

Scansionato con CamScanner

Classe ClientStd

```
public class ClientStd {
    private static PezzoEJBRemote ejb;
    public static void main (String[] args)
                        throws NamingException {
        Context ctx = new InitialContext();
        ejb = (PezzoEJBRemote)
            ctx.lookup("java:global/CIRILLOANTONIO_EJB/"
                + "PezzoEJB!it.unisa.PezzoEJBRemote");
        printAll();
        printByCategory(Categorie.MOTORE);
        printToPurchase();
    }

    public static void printAll() {
        ejb.ottieniTutti().forEach(
            (p) -> System.out.println(p);
        );
        System.out.println();
    }
```

Scansionato con CamScanner

```
public static void printByCategory(Categorie category) {
    ejb.ottieniDaCategorie(category).forEach(
        (p) -> System.out.println(p)
    );
    System.out.println();
}

public static void printToPurchase() {
    ejb.ottieniDaComprare().forEach(
        (p) -> System.out.println(p)
    );
    System.out.println();
}

}
```

STRUTTURA PROGETTO STDCLIENT, NOME: CIRILLO ANTONIO_STDCLIENT

SRC
- ↳ it.unisa
  - ↳ PETTOENTITY
  - PETTOEJBREMOTE
  - CLIENTSTD

LIBRARY
- ↳ GF-CLIENT
- JAVA-EE 8 API

Scansionato con CamScanner

Classe ClientSMS

```
public class ClientSMS {
    public static void main (String [] args) throws NamingException {
        Context ctx = new InitialContext();
        ConnectionFactory cf = (ConnectionFactory)
            ctx.lookup("jms/jonas7/ConnectionFactory");
        Destination topic = (Destination)
            ctx.lookup("jms/jonas7/Topic");

        Scanner in = new Scanner(System.in);
        int id = in.nextInt();
        int vendite = in.nextInt();
        try (JMSContext jmsCtx = cf.createContext()) {
            jmsContext.createProducer()
                    .setProperty("id", id)
                    .send(topic, vendite);
        }
    }
}
```

Scansionato con CamScanner

STRUTTURA PROGETTO JTS CLIENT, NOME PROGETTO:

CIRILLO ANTONIO_JTS CLIENT

SRC
↳ it.unisa
↳ CLIENT JTS

LIBRARY
↳ JAVA EE 8 API
GF-CLIENT

Classe ClientWS

```
public class ClientWS {
    public static void main (String[] args) {
        Scanner in = new Scanner (System.in);
        int id = in.nextInt()
        double price = in.nextDouble();
        System.out.println(updatePrice(id, price));
    }

    public static it.unisa.PeccoEntity updatePrice(int id, double price) {
        it.unisa.PeccoESBService service = it.unisa.PeccoESBService();
        it.unisa.PeccoESB port = service.getPeccoESBPort();
        return port.updatePrice(id, price);
    }
}
```

Scansionato con CamScanner

STRUTTURA PROGETTO WSCLIEN, NOTE PROGETTO:

GRILLO ANTONIO_WSCLIENT

Bisogna creare un Web Service Client e impostare come WSDL il progetto

GRILLO ANTONIO_ESB

SRC

↳ it.unisa.

↳ CLIENT WS

LIBRARY

↳ JAVA EE 8 API

Scansionato con CamScanner