# JPA Entity implementation:

The implementation of our application includes three JPA entity. In this section we are going to explain the implementation of this fundamental part, in particular the in-line code annotation used.

* **Entity PC:**

1. @Entity
2. public class PC implements Serializable {
4. @Id **[1]**
5. @GeneratedValue(strategy = GenerationType.AUTO) **[2]**
6. @Column(name="pcId") **[3]**
7. private long pcId;
9. @Column(name="pcNumber")**[4]**
10. private int pcNumber;
12. @ManyToOne **[5]**
13. @JoinColumn(name = "pcRoom") **[6]**
14. private Room pcRoom;
16. @OneToMany(mappedBy="pcBooked") **[7]**
17. private Set<Reservation> reservations;
19. //Setter and getter
20. //…
21. }
    1. **@Id** → This annotation is used to point that the attribute pcId will be the primary key of the table.
    2. **@GeneratedValue(strategy = GenerationType.AUTO)** → This annotation specify how to generate the value of pcId. The *GenerationType.A*UTO is the default generation type and lets the persistence provider choose the generation strategy.
    3. **@Column(name="pcId")** → This annotation is used to mention the details of the column related to the attribute pcId in particular the name of this column will be pcId.
    4. **@Column(name="pcNumber")** → has already said above with this annotation we set the column name as pcNumber.
    5. **@ManyToOne** → between the entities Room and PC exists a many to one relation, in facts every PC is associated to one and only one Room and a Room can hold to one or more PC. The *@ManyToOne* annotation is used to create the many-to-one relationship between the Room and PC entities.
    6. **@JoinColumn** → This annotation is used to specify the column for joying an entity association. This column indicate that this entity is the owner of the relationship. It’s also used to specify the name of the column.
    7. **@OneToMany(mappedBy="pcBooked")** → Between PC and Reservation entities exist a many to one relation, in facts every Reservation refer to a PC and a PC can be referred by one or more Reservation. The attribute mappedBy indicates that the entity in this side is the inverse of the relationship, and the owner resides in the "other" entity.

* **Entity Room:**

1. @Entity
2. public class Room implements Serializable {
4. @Id **[1]**
5. @Column(name = "roomName")**[2]**
6. private String roomName;
8. @Column(name = "capacity")**[3]**
9. private int capacity;
11. @Column(name = "rowsNumber")**[4]**
12. private int rowsNumber;
14. @OneToMany(mappedBy = "pcRoom")**[5]**
15. private Set<PC> PCs;
17. @Transient **[6]**
18. private int availablePCs;
20. //Getter and setter
21. //…
22. }

* 1. **@Id** → This annotation is used to point that the attribute roomName will be the primary key of the table.
  2. **@Column(name="** **roomName ")** → This annotation is used to mention the details of the column related to the attribute roomName in particular the name of this column will be roomName.
  3. **@Column(name="** **capacity ")** → as above this annotation is used to mention the details of the column related to the attribute capacity in particular the name of this column will be capacity.
  4. **@Column(name="** **rowsNumber ")** → again this annotation is used to mention the details of the column related to the attribute roomName in particular the name of this column will be roomName.
  5. **@OneToMany(mappedBy="pcRoom")** → as said before between the entities PC and Room exist a many to one relation that is own by the entity PC and so we use the attributed mapped by = “pcRoom”.
  6. **@Transient** → this annotation is used to indicate that the field availablePCs is not be persisted in the database. We use this transient field in order to present the room’s data in the table view inside the application.
* **Entity Reservation:**

1. @Entity
2. public class Reservation implements Serializable {
4. @Id **[1]**
5. @Column(name = "startTime") **[2]**
6. private String startTime;
7. @Id **[3]**
8. @Column(name = "bookingDate") **[4]**
9. private String bookingDate;
10. @Id **[5]**
11. @ManyToOne **[6]**
12. private PC pcBooked;
14. @Transient **[7]**
15. private String roomN;
17. @Transient **[8]**
18. private int pcnumb;
20. @Column(name = "username") **[9]**
21. private String username;
23. //Getter and setter
24. //…
26. }
    1. **@Id** → This annotation is used to point that the attribute startTime will be a part of the primary key of the table.
    2. **@Column(name="** **startTime ")** → This annotation is used to mention the details of the column related to the attribute startTime in particular the name of this column will be startTime.
    3. **@Id** → This annotation is used to point that the attribute bookingDate will be a part of the primary key of the table.
    4. **@Column(name="** **startTime ")** → This annotation is used to mention the details of the column related to the attribute startTime in particular the name of this column will be startTime.
    5. **@Id** → This annotation is used to point that the attribute pcBooked will be a part of the primary key of the table
    6. **@ManyToOne** → as explained before between the entities Reservation and PC exists a many to one relation. The *@ManyToOne* annotation is used to create the many-to-one relationship between the Reservation and PC entities that is own by this entity.
    7. **@Transient** → as said above this annotation is used to indicate that the field roomN is not be persisted in the database. We use this transient field in order to present the PC’s data associated to the Resevation in the table view inside the application.
    8. **@Transient** → again this annotation is used to indicate that the field pcnumb is not be persisted in the database. As above we use this transient field in order to present the PC’s data associated to the Resevation in the table view inside the application.
    9. **@Column(name="** **username ")** → This annotation is used to mention the details of the column related to the attribute username in particular the name of this column will be username.