Project of Data base 2

A.A. 2021/2022
Computer Science Engineering
By: Valentina Politi and Martino Manzolini

Consumer Application

What we can do:

- Create Orders
- Pay rejected orders
- View packages
- View Optional products
- View all orders

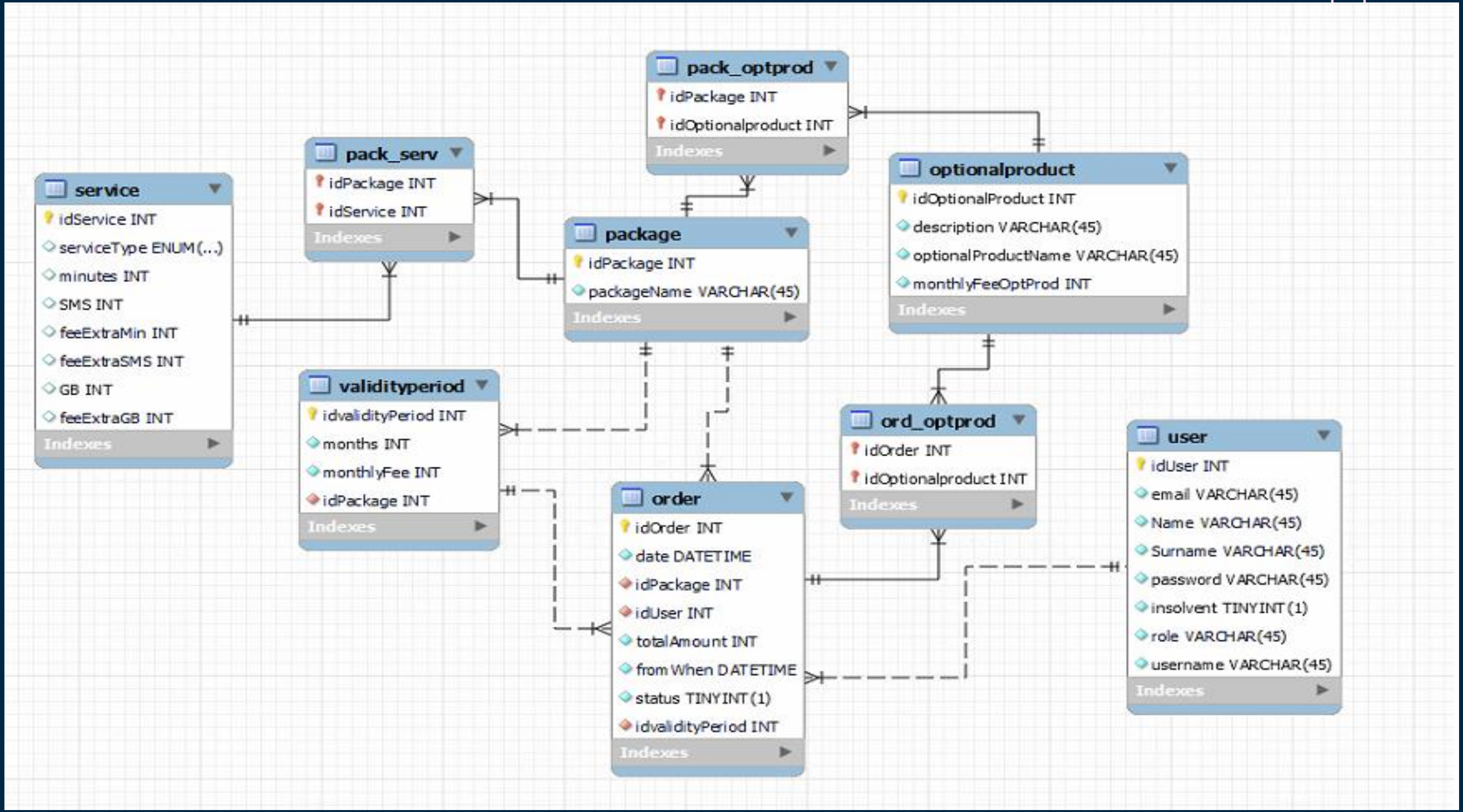
Employee Application

What we can do:

- Create new packages
- Create new optional products
- View following statistics of sales:

  - ◉ Number of total purchases per package.
  - ◉ Number of total purchases per package and validity period.
  - ◉ Total value of sales per package with and without the optional products.
  - ◉ Average number of optional products sold together with each service package.
  - ◉ List of insolvent users, suspended orders and alerts.
  - ◉ Best seller optional product, i.e. the optional product with the greatest value of sales across all the sold service packages.

Relational Model Diagram



pack_optprod
- idPackage INT
- idOptionalproduct INT
- Indexes

pack_serv
- idPackage INT
- idService INT
- Indexes

service
- idService INT
- serviceType ENUM(...)
- minutes INT
- SMS INT
- feeExtraMin INT
- feeExtraSMS INT
- GB INT
- feeExtraGB INT
- Indexes

package
- idPackage INT
- packageName VARCHAR(45)
- Indexes

optionalproduct
- idOptionalProduct INT
- description VARCHAR(45)
- optionalProductName VARCHAR(45)
- monthlyFeeOptProd INT
- Indexes

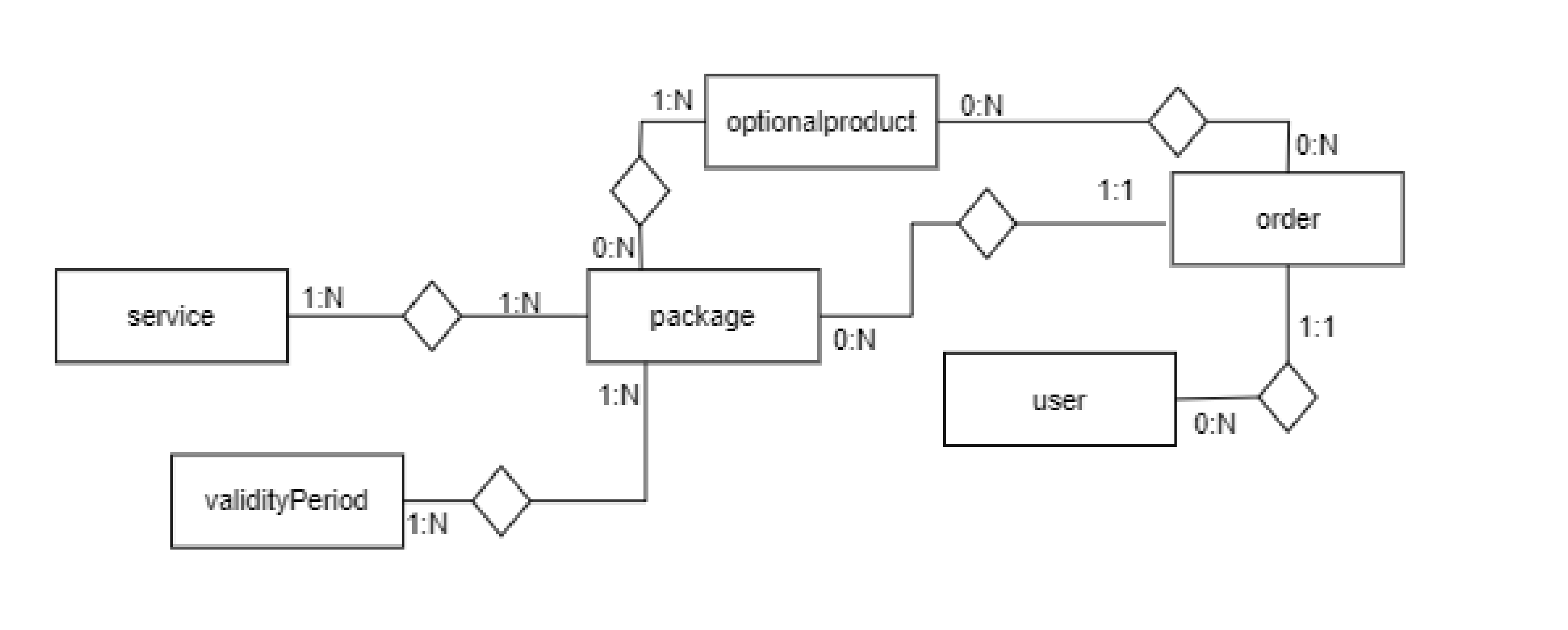
validityperiod
- idvalidityPeriod INT
- months INT
- monthlyFee INT
- idPackage INT
- Indexes

ord_optprod
- idOrder INT
- idOptionalproduct INT
- Indexes

order
- idOrder INT
- date DATETIME
- idPackage INT
- idUser INT
- totalAmount INT
- fromWhen DATETIME
- status TINYINT(1)
- idvalidityPeriod INT
- Indexes

user
- idUser INT
- email VARCHAR(45)
- Name VARCHAR(45)
- Surname VARCHAR(45)
- password VARCHAR(45)
- insolvent TINYINT(1)
- role VARCHAR(45)
- username VARCHAR(45)
- Indexes

Conceptual and Logical Data Model

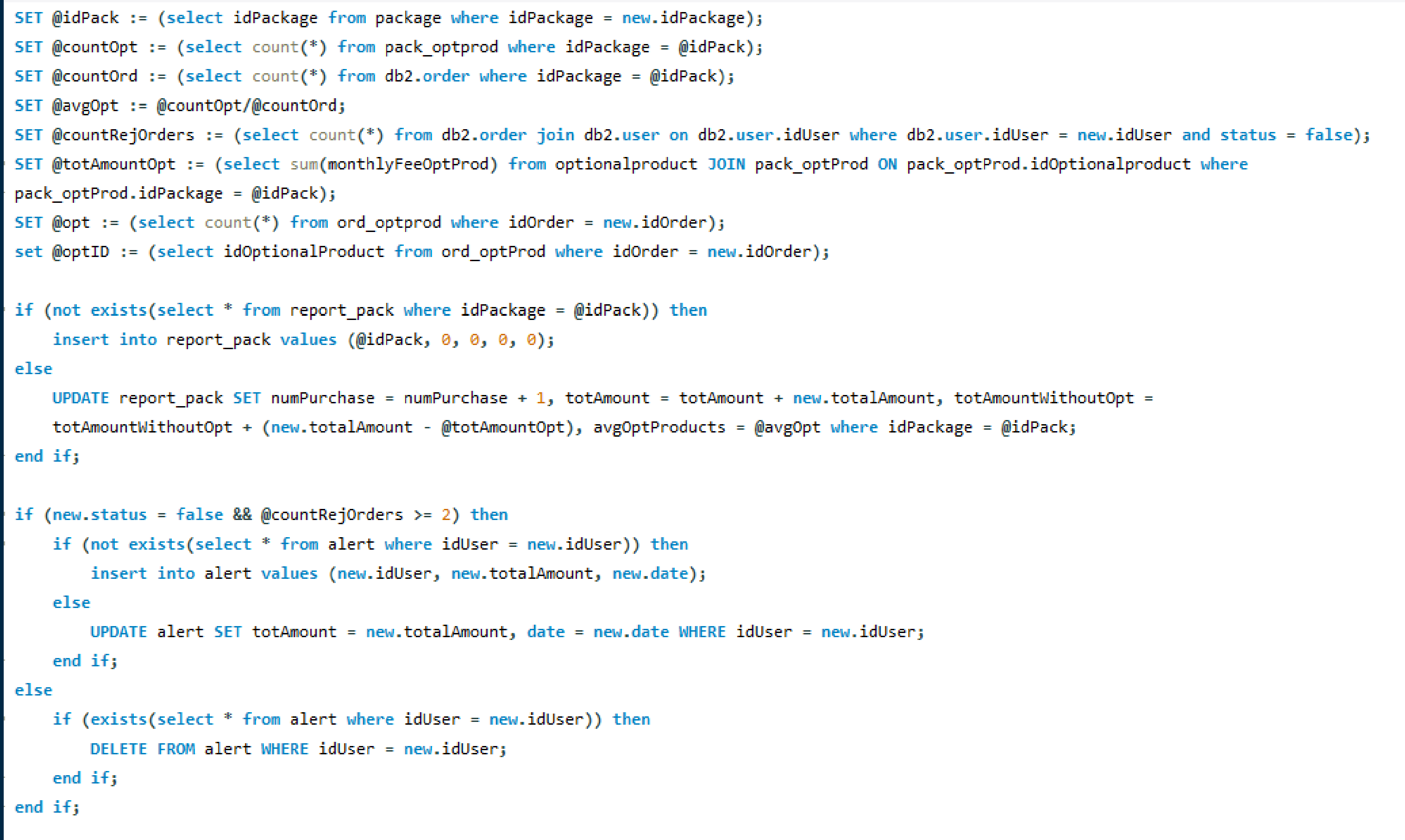


Trigger design & code

We created 5 triggers to maintain the materialized views that are not supported in MySQL.

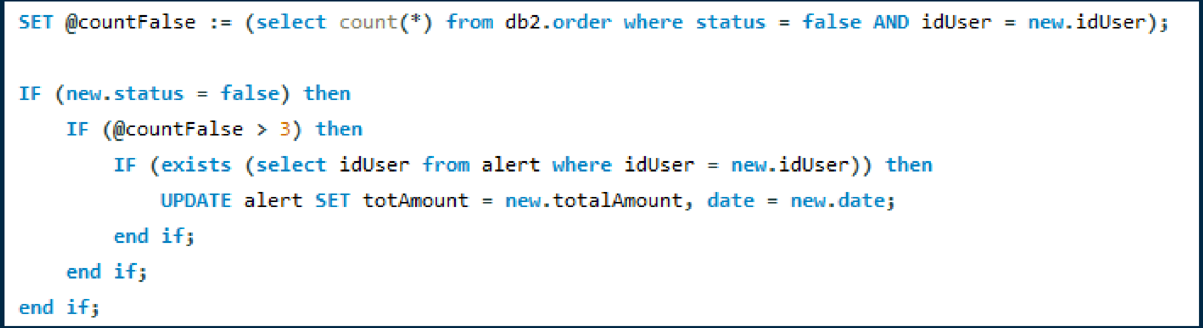
- A trigger that, after the insertion of a new order, create/update the correspond record in the report_pack, report_pack_validity and report_optprod with new values needed to the sales report page. It also check if the user have more then 3 rejected orders, and if it is, it creates/updates the correspond record in the alert table.
- A trigger that, after the update of an already exists order (such as a new temptative of payment), as before make the check of the user if it should be present in the alert table.
- 3 triggers that, after the insertion of new OptionalProduct, Package and ValidityPeriod, creates the correspond record on the related report table.

After insert of Order



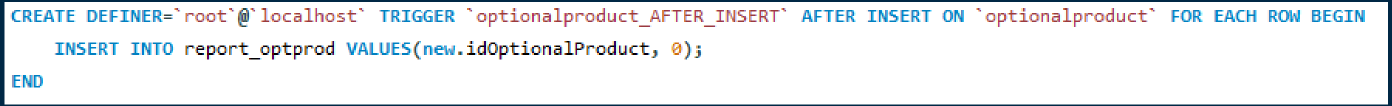
```
SET @idPack := (select idPackage from package where idPackage = new.idPackage);
SET @countOpt := (select count(*) from pack_optprod where idPackage = @idPack);
SET @countOrd := (select count(*) from db2.order where idPackage = @idPack);
SET @avgOpt := @countOpt/@countOrd;
SET @countRejOrders := (select count(*) from db2.order join db2.user on db2.user.idUser where db2.user.idUser = new.idUser and status = false);
SET @totAmountOpt := (select sum(monthlyFeeOptProd) from optionalproduct JOIN pack_optProd ON pack_optProd.idOptionalproduct where
pack_optProd.idPackage = @idPack);
SET @opt := (select count(*) from ord_optprod where idOrder = new.idOrder);
set @optID := (select idOptionalProduct from ord_optProd where idOrder = new.idOrder);

if (not exists(select * from report_pack where idPackage = @idPack)) then
    insert into report_pack values (@idPack, 0, 0, 0, 0);
else
    UPDATE report_pack SET numPurchase = numPurchase + 1, totAmount = totAmount + new.totalAmount, totAmountWithoutOpt =
    totAmountWithoutOpt + (new.totalAmount - @totAmountOpt), avgOptProducts = @avgOpt where idPackage = @idPack;
end if;

if (new.status = false && @countRejOrders >= 2) then
    if (not exists(select * from alert where idUser = new.idUser)) then
        insert into alert values (new.idUser, new.totalAmount, new.date);
    else
        UPDATE alert SET totAmount = new.totalAmount, date = new.date WHERE idUser = new.idUser;
    end if;
else
    if (exists(select * from alert where idUser = new.idUser)) then
        DELETE FROM alert WHERE idUser = new.idUser;
    end if;
end if;
```

After update Order status



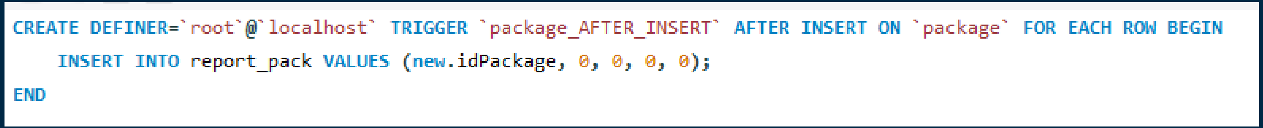
```
SET @countFalse := (select count(*) from db2.order where status = false AND idUser = new.idUser);

IF (new.status = false) then
    IF (@countFalse > 3) then
        IF (exists (select idUser from alert where idUser = new.idUser)) then
            UPDATE alert SET totAmount = new.totalAmount, date = new.date;
        end if;
    end if;
end if;
```

After insert OptionalProducts



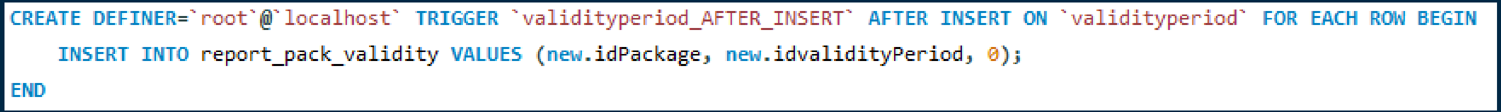
```
CREATE DEFINER=`root`@`localhost` TRIGGER `optionalproduct_AFTER_INSERT` AFTER INSERT ON `optionalproduct` FOR EACH ROW BEGIN
    INSERT INTO report_optprod VALUES(new.idOptionalProduct, 0);
END
```

After insert Package



```
CREATE DEFINER=`root`@`localhost` TRIGGER `package_AFTER_INSERT` AFTER INSERT ON `package` FOR EACH ROW BEGIN
    INSERT INTO report_pack VALUES (new.idPackage, 0, 0, 0, 0);
END
```

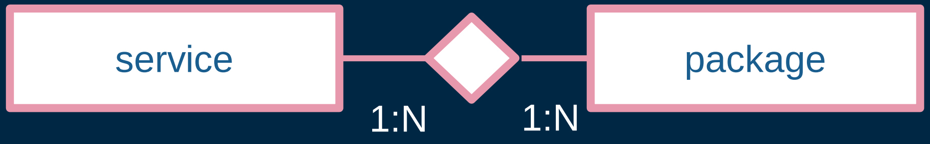
After insert ValidityPeriod

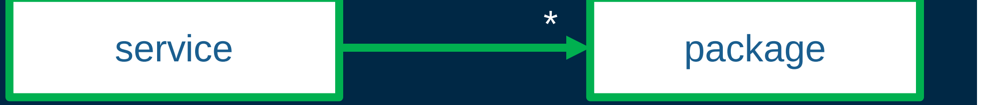


```
CREATE DEFINER=`root`@`localhost` TRIGGER `validityperiod_AFTER_INSERT` AFTER INSERT ON `validityperiod` FOR EACH ROW BEGIN
    INSERT INTO report_pack_validity VALUES (new.idPackage, new.idvalidityPeriod, 0);
END
```

ORM design

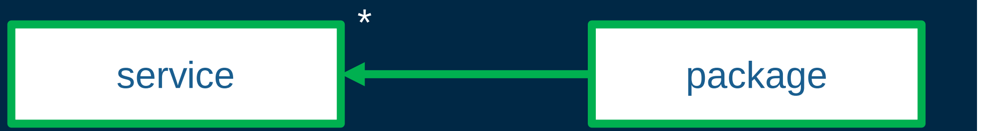
Relationship service-package





service → package

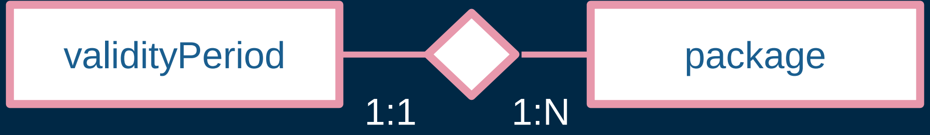
```
@ManyToMany

@JoinTable(name = "pack_serv", schema = "db2",
joinColumns = @JoinColumn(name = "idService"),
inverseJoinColumns = @JoinColumn(name = "idPackage"))

private List<Package> packages;
```

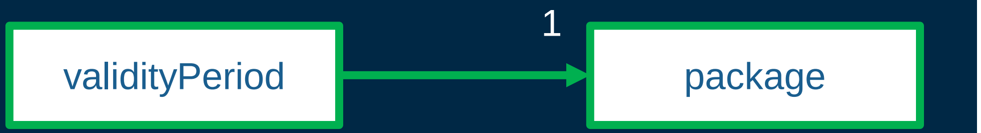


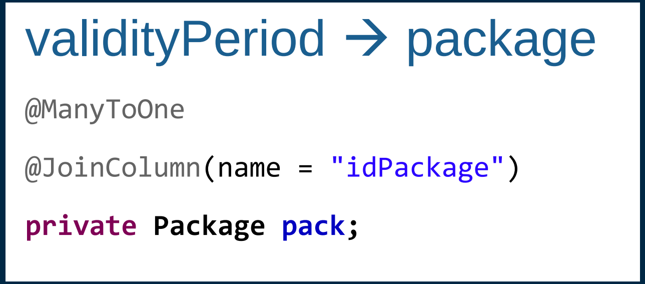
package → service

```
@ManyToMany(mappedBy="packages")
private List<Service> services;
```

Relationship validityPeriod-package

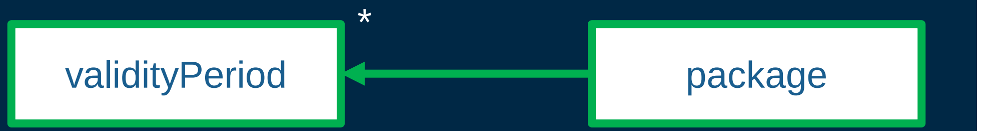


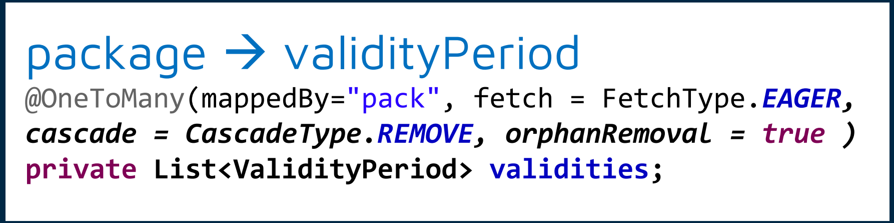




validityPeriod → package

```
@ManyToOne
@JoinColumn(name = "idPackage")
private Package pack;
```

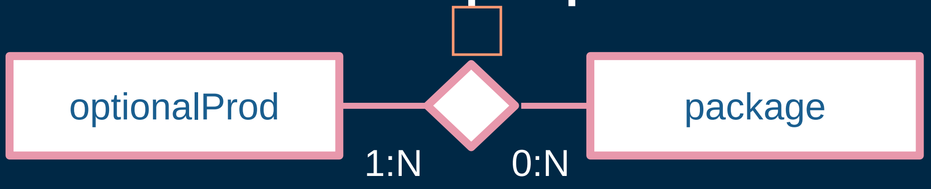


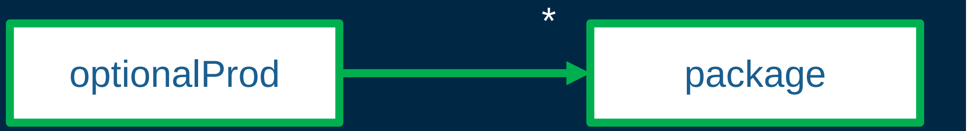


package → validityPeriod

```
@OneToMany(mappedBy="pack", fetch = FetchType.EAGER,
cascade = CascadeType.REMOVE, orphanRemoval = true )
private List<ValidityPeriod> validities;
```

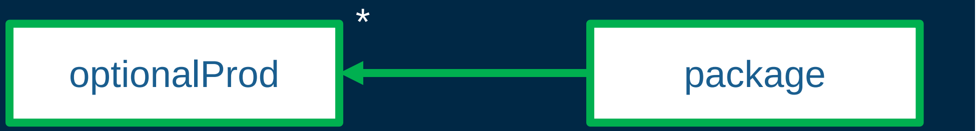
Relationship optionalProd-package





optionalProd → package

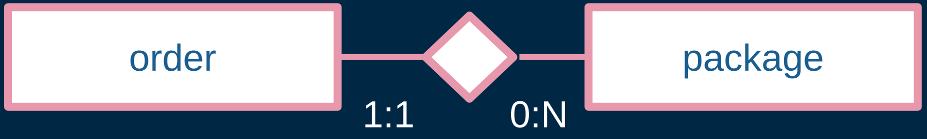
```
@ManyToMany

@JoinTable(name = "pack_optprod", schema = "db2",
joinColumns = @JoinColumn(name = "idOptionalproduct"),
inverseJoinColumns = @JoinColumn(name = "idPackage"))

private List<Package> pack;
```

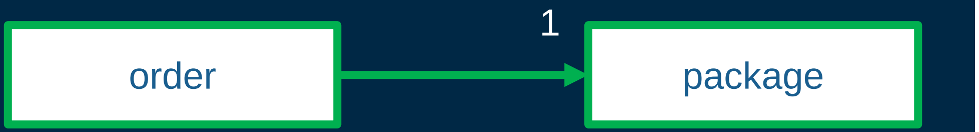


package → optionalProd

```
@ManyToMany(mappedBy="pack")
private List<OptionalProduct> optProducts;
```

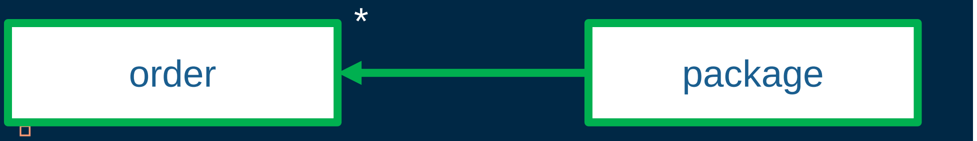
Relationship order-package





order → package

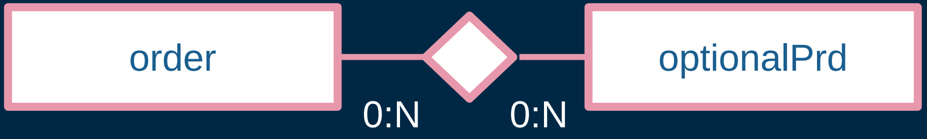
```
@ManyToOne
@JoinColumn(name = "idPackage")
private Package pack;
```



package → order

```
@OneToMany(mappedBy="pack", fetch = FetchType.EAGER,
cascade = CascadeType.REMOVE, orphanRemoval = true )
private List<Order> orders;
```

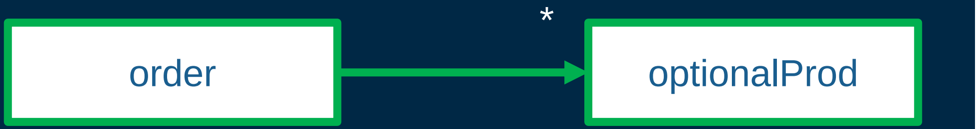
Relationship order-optionalProd


order

optionalPrd

0:N

0:N

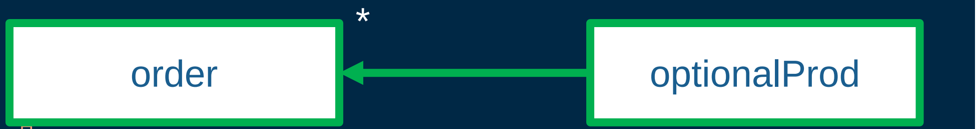

order

optionalProd

*

order → optionalProd

```
@ManyToMany

@JoinTable(name = "ord_optprod", schema = "db2",
joinColumns = @JoinColumn(name = "idOrder"),
inverseJoinColumns = @JoinColumn(name =
"idOptionalproduct"))

private List<OptionalProduct> optProducts;
```


order

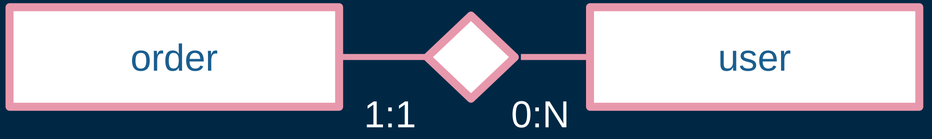
optionalProd

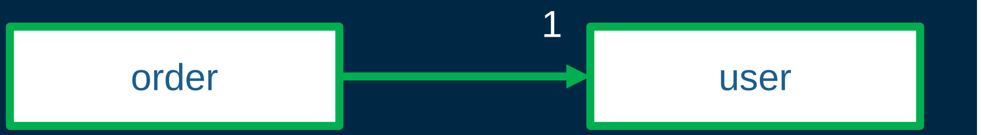
*

optionalProd → order

```
@ManyToMany
@JoinTable(name = "ord-optprod", schema = "db2",
joinColumns = @JoinColumn(name =
"idOptionalproduct"), inverseJoinColumns =
@JoinColumn(name = "idOrder"))
private List<Order> order;
```

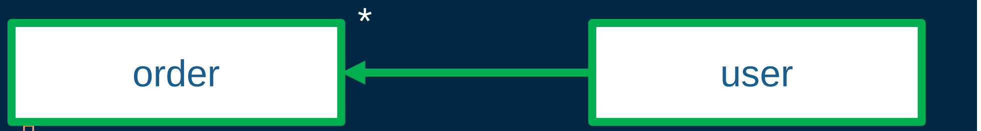
Relationship order-user





order → user

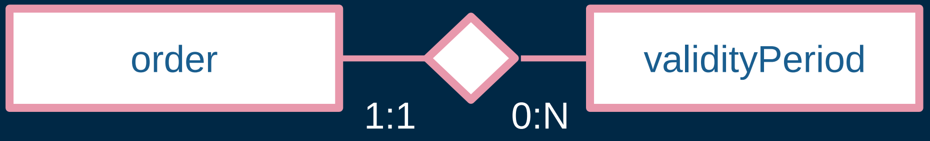
```
@ManyToOne
@JoinColumn(name = "idUser")
private User user;
```

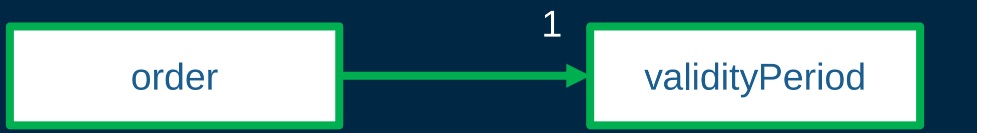


user → order

```
@OneToMany(mappedBy="user", fetch = FetchType.EAGER,
cascade = CascadeType.REMOVE, orphanRemoval = true )
private List<Order> orders;
```

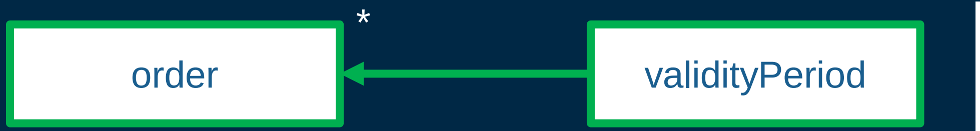
Relationship order-validityPeriod





order → validityPeriod

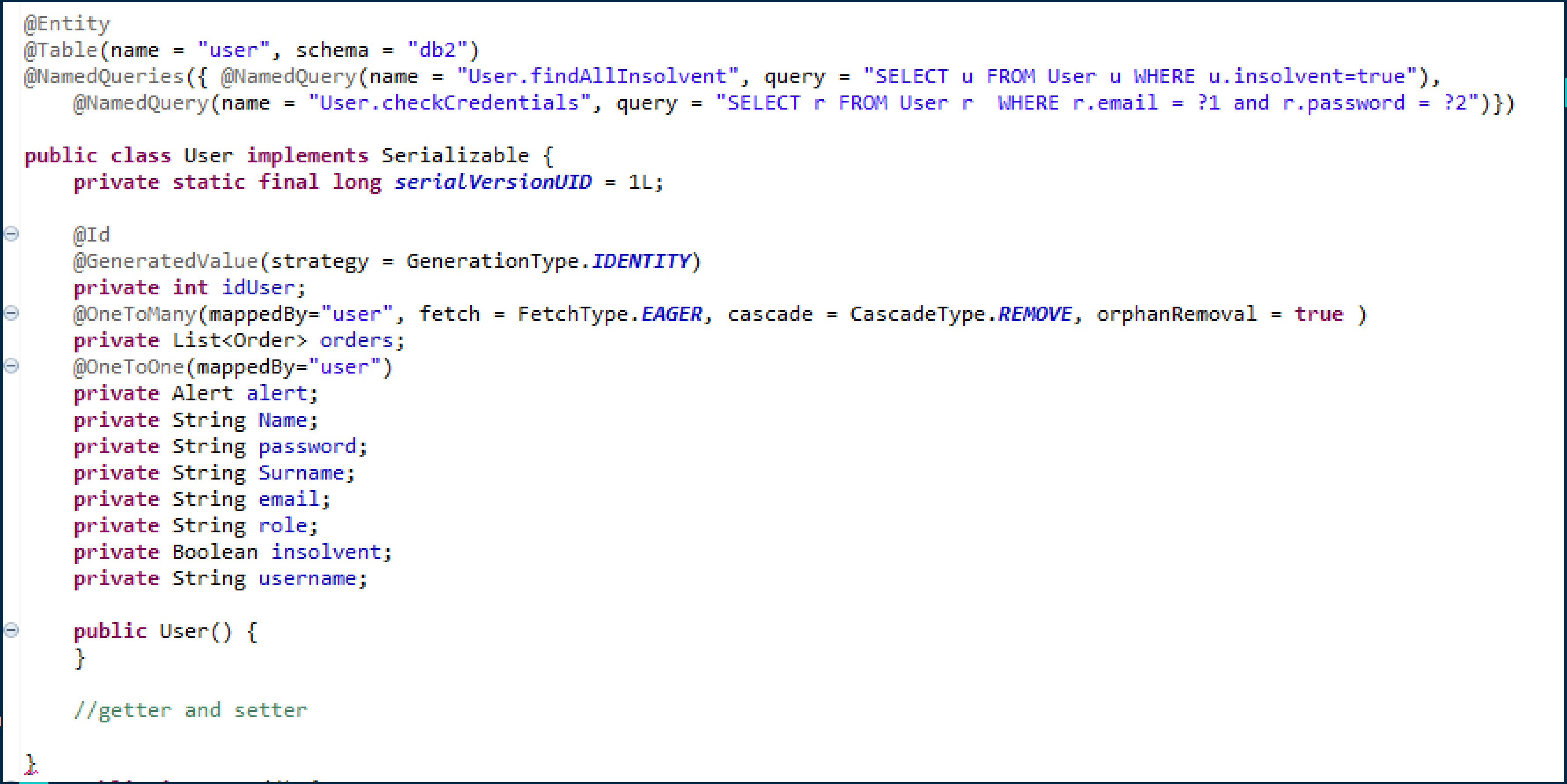
```
@ManyToOne
@JoinColumn(name = "idvalidityPeriod")
private ValidityPeriod period;
```



validityPeriod → order

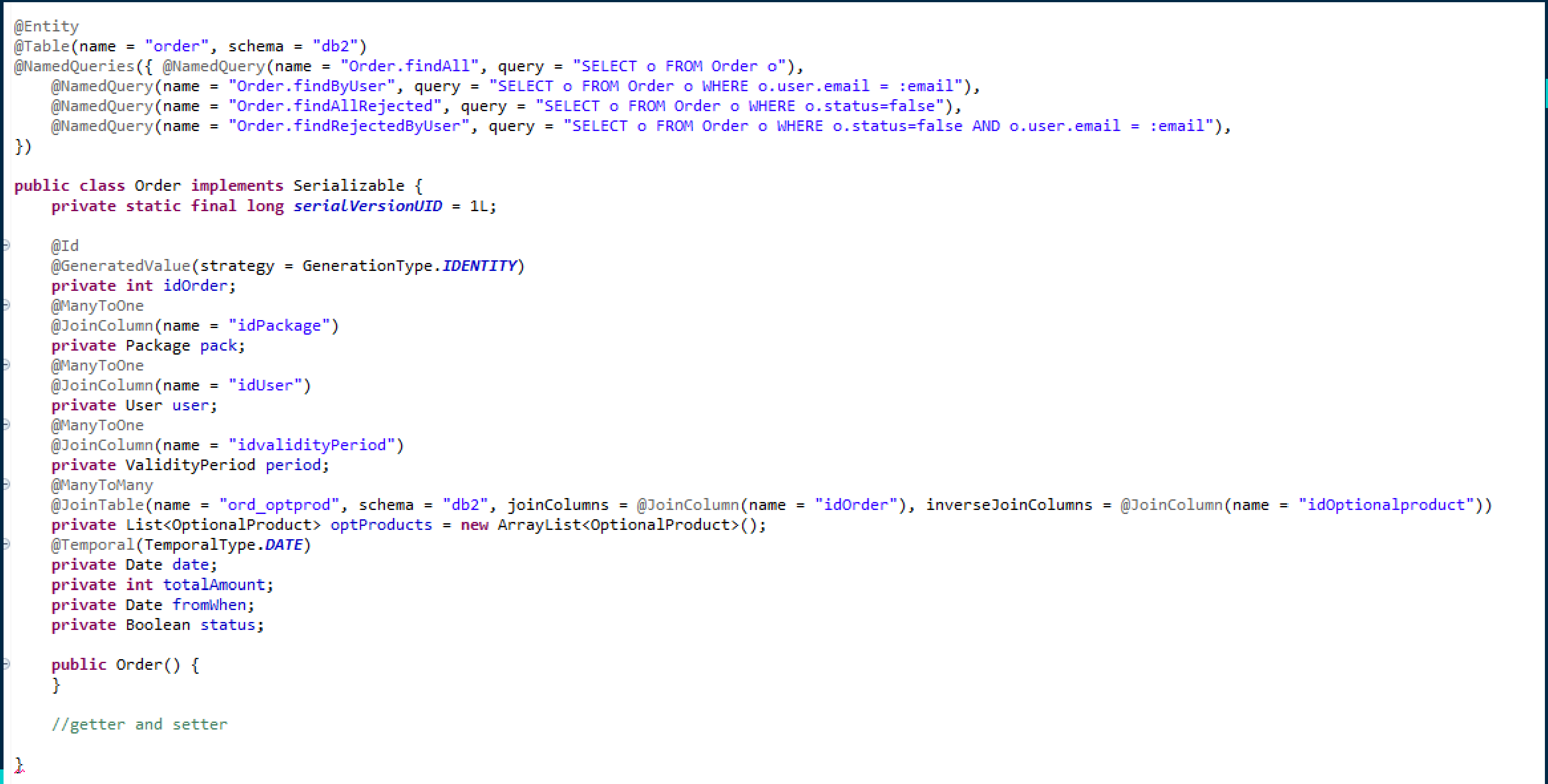
No mapped

Entity User



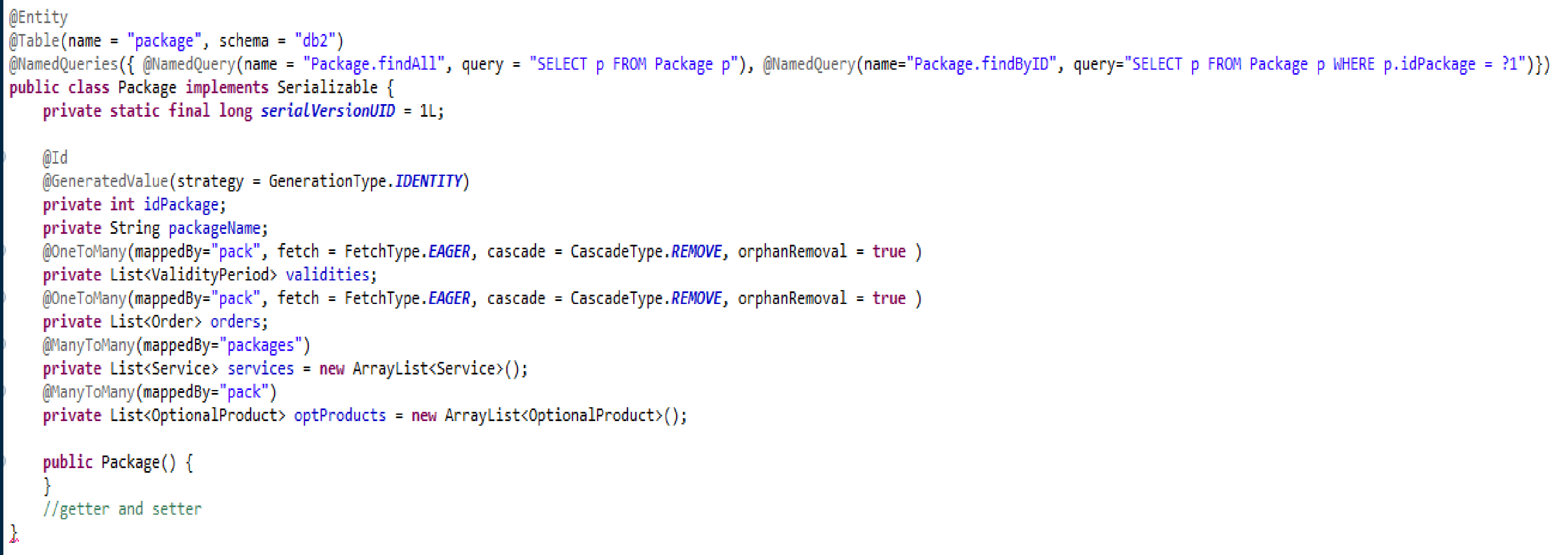
```
@Entity
@Table(name = "user", schema = "db2")
@NamedQueries({ @NamedQuery(name = "User.findAllInsolvent", query = "SELECT u FROM User u WHERE u.insolvent=true"),
    @NamedQuery(name = "User.checkCredentials", query = "SELECT r FROM User r  WHERE r.email = ?1 and r.password = ?2")})

public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idUser;
    @OneToMany(mappedBy="user", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE, orphanRemoval = true )
    private List<Order> orders;
    @OneToOne(mappedBy="user")
    private Alert alert;
    private String Name;
    private String password;
    private String Surname;
    private String email;
    private String role;
    private Boolean insolvent;
    private String username;

    public User() {
    }

    //getter and setter

}
```

Entity Order



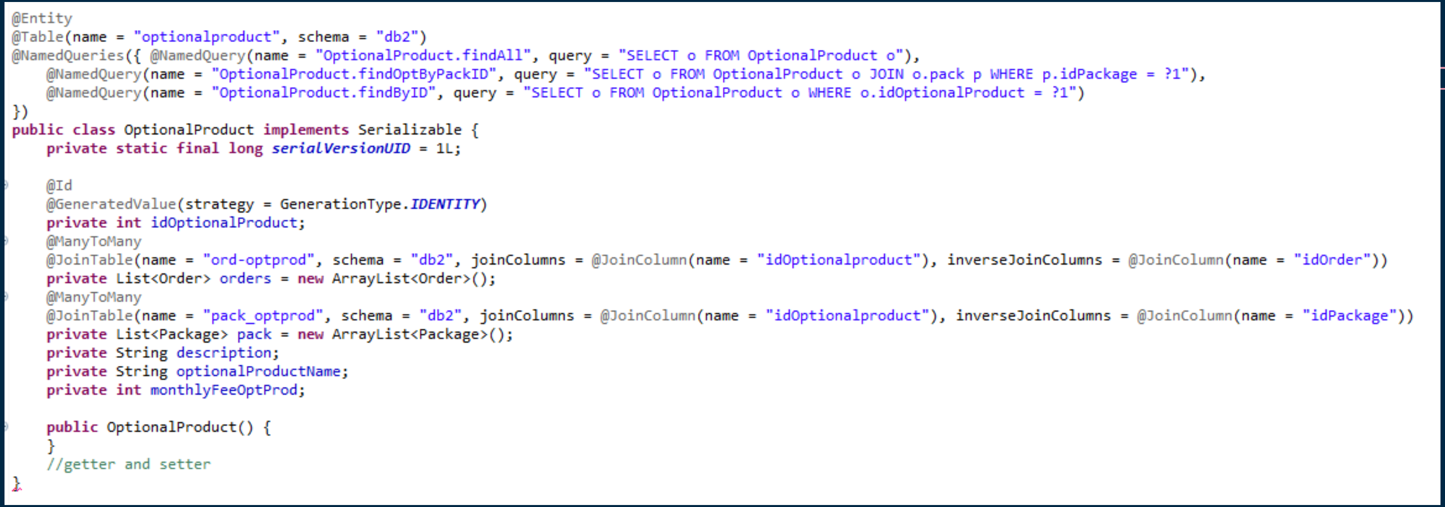
```
@Entity
@Table(name = "order", schema = "db2")
@NamedQueries({ @NamedQuery(name = "Order.findAll", query = "SELECT o FROM Order o"),
    @NamedQuery(name = "Order.findByUser", query = "SELECT o FROM Order o WHERE o.user.email = :email"),
    @NamedQuery(name = "Order.findAllRejected", query = "SELECT o FROM Order o WHERE o.status=false"),
    @NamedQuery(name = "Order.findRejectedByUser", query = "SELECT o FROM Order o WHERE o.status=false AND o.user.email = :email"),
})

public class Order implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idOrder;
    @ManyToOne
    @JoinColumn(name = "idPackage")
    private Package pack;
    @ManyToOne
    @JoinColumn(name = "idUser")
    private User user;
    @ManyToOne
    @JoinColumn(name = "idvalidityPeriod")
    private ValidityPeriod period;
    @ManyToMany
    @JoinTable(name = "ord_optprod", schema = "db2", joinColumns = @JoinColumn(name = "idOrder"), inverseJoinColumns = @JoinColumn(name = "idOptionalproduct"))
    private List<OptionalProduct> optProducts = new ArrayList<OptionalProduct>();
    @Temporal(TemporalType.DATE)
    private Date date;
    private int totalAmount;
    private Date fromWhen;
    private Boolean status;

    public Order() {
    }

    //getter and setter

}
```

Entity Package



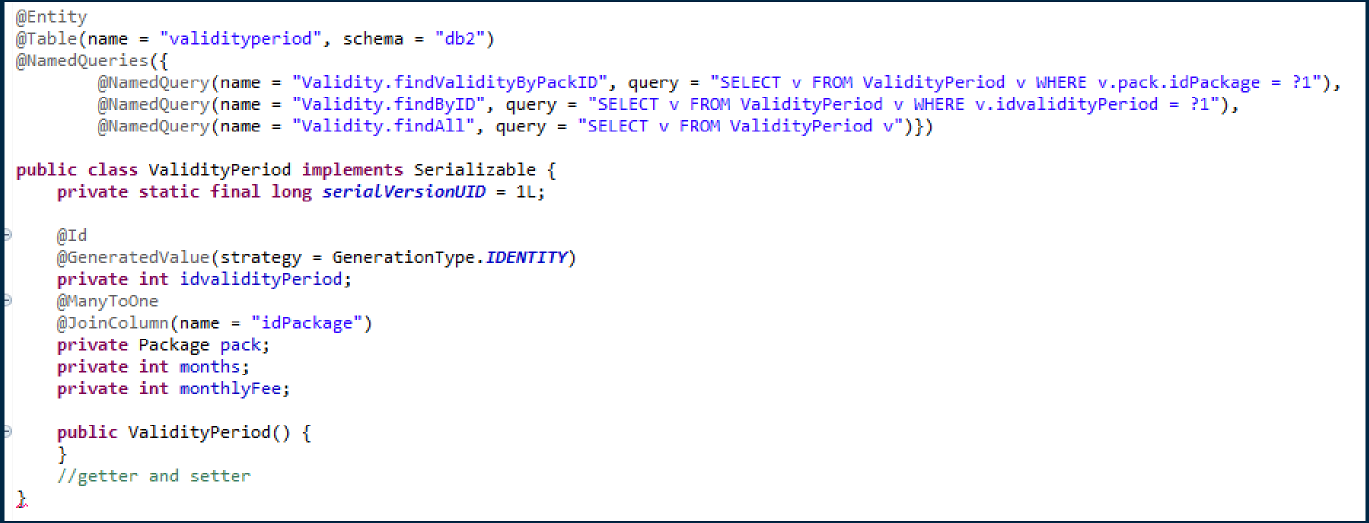
```java
@Entity
@Table(name = "package", schema = "db2")
@NamedQueries({ @NamedQuery(name = "Package.findAll", query = "SELECT p FROM Package p"), @NamedQuery(name="Package.findByID", query="SELECT p FROM Package p WHERE p.idPackage = ?1")})
public class Package implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idPackage;
    private String packageName;
    @OneToMany(mappedBy="pack", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE, orphanRemoval = true )
    private List<ValidityPeriod> validities;
    @OneToMany(mappedBy="pack", fetch = FetchType.EAGER, cascade = CascadeType.REMOVE, orphanRemoval = true )
    private List<Order> orders;
    @ManyToMany(mappedBy="packages")
    private List<Service> services = new ArrayList<Service>();
    @ManyToMany(mappedBy="pack")
    private List<OptionalProduct> optProducts = new ArrayList<OptionalProduct>();

    public Package() {
    }
    //getter and setter
}
```

Entity OptionalProduct

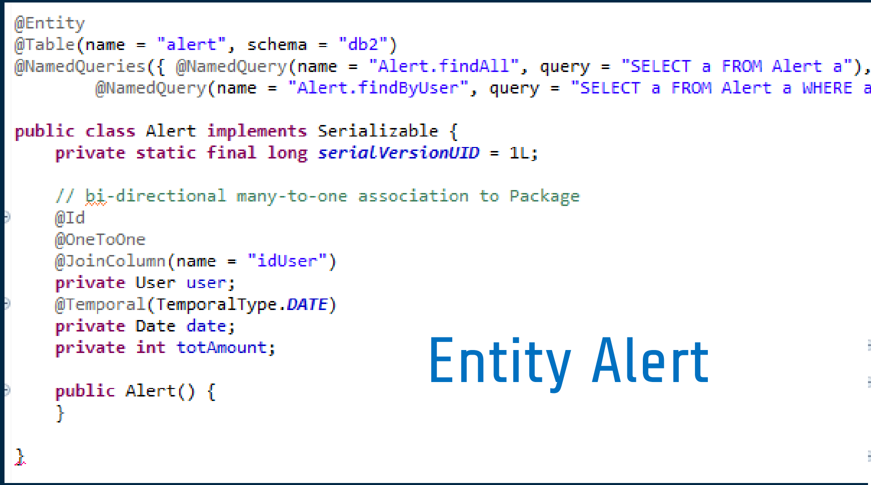


```java
@Entity
@Table(name = "optionalproduct", schema = "db2")
@NamedQueries({ @NamedQuery(name = "OptionalProduct.findAll", query = "SELECT o FROM OptionalProduct o"),
    @NamedQuery(name = "OptionalProduct.findOptByPackID", query = "SELECT o FROM OptionalProduct o JOIN o.pack p WHERE p.idPackage = ?1"),
    @NamedQuery(name = "OptionalProduct.findByID", query = "SELECT o FROM OptionalProduct o WHERE o.idOptionalProduct = ?1")
})
public class OptionalProduct implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idOptionalProduct;
    @ManyToMany
    @JoinTable(name = "ord-optprod", schema = "db2", joinColumns = @JoinColumn(name = "idOptionalproduct"), inverseJoinColumns = @JoinColumn(name = "idOrder"))
    private List<Order> orders = new ArrayList<Order>();
    @ManyToMany
    @JoinTable(name = "pack_optprod", schema = "db2", joinColumns = @JoinColumn(name = "idOptionalproduct"), inverseJoinColumns = @JoinColumn(name = "idPackage"))
    private List<Package> pack = new ArrayList<Package>();
    private String description;
    private String optionalProductName;
    private int monthlyFeeOptProd;

    public OptionalProduct() {
    }
    //getter and setter
}
```

Entity ValidityPeriod

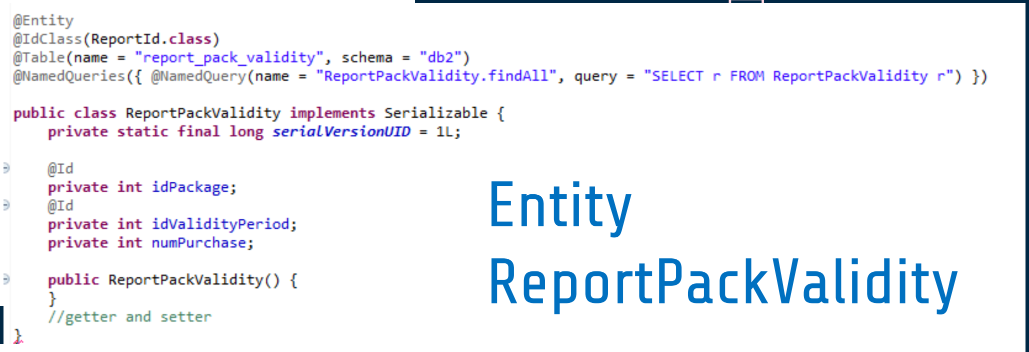


```java
@Entity
@Table(name = "validityperiod", schema = "db2")
@NamedQueries({
        @NamedQuery(name = "Validity.findValidityByPackID", query = "SELECT v FROM ValidityPeriod v WHERE v.pack.idPackage = ?1"),
        @NamedQuery(name = "Validity.findByID", query = "SELECT v FROM ValidityPeriod v WHERE v.idvalidityPeriod = ?1"),
        @NamedQuery(name = "Validity.findAll", query = "SELECT v FROM ValidityPeriod v")})

public class ValidityPeriod implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idvalidityPeriod;
    @ManyToOne
    @JoinColumn(name = "idPackage")
    private Package pack;
    private int months;
    private int monthlyFee;

    public ValidityPeriod() {
    }
    //getter and setter
}
```



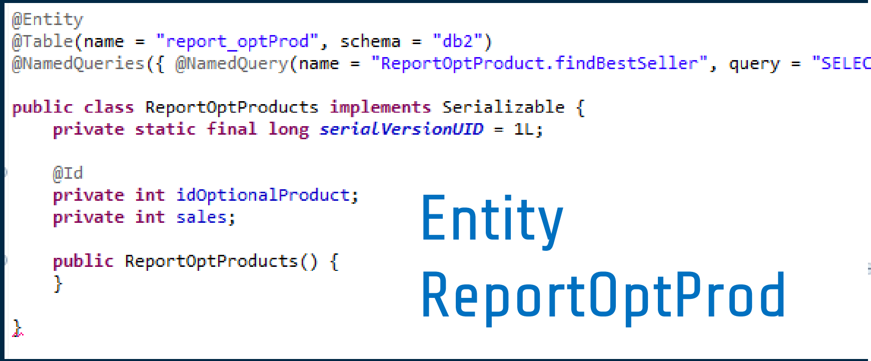
```
@Entity
@Table(name = "alert", schema = "db2")
@NamedQueries({ @NamedQuery(name = "Alert.findAll", query = "SELECT a FROM Alert a"),
        @NamedQuery(name = "Alert.findByUser", query = "SELECT a FROM Alert a WHERE a.user.idUser = ?1"), })

public class Alert implements Serializable {
    private static final long serialVersionUID = 1L;

    // bi-directional many-to-one association to Package
    @Id
    @OneToOne
    @JoinColumn(name = "idUser")
    private User user;
    @Temporal(TemporalType.DATE)
    private Date date;
    private int totAmount;

    public Alert() {
    }

}
```

Entity Alert



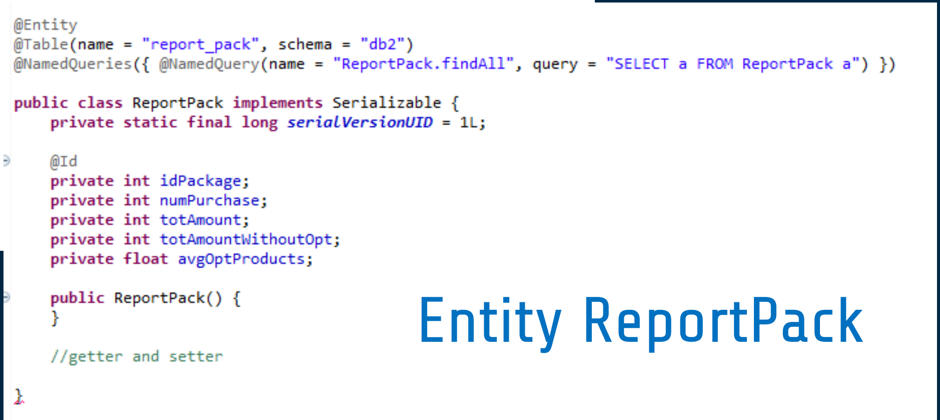
```
@Entity
@IdClass(ReportId.class)
@Table(name = "report_pack_validity", schema = "db2")
@NamedQueries({ @NamedQuery(name = "ReportPackValidity.findAll", query = "SELECT r FROM ReportPackValidity r") })

public class ReportPackValidity implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    private int idPackage;
    @Id
    private int idValidityPeriod;
    private int numPurchase;

    public ReportPackValidity() {
    }
    //getter and setter
}
```

Entity ReportPackValidity



```
@Entity
@Table(name = "report_optProd", schema = "db2")
@NamedQueries({ @NamedQuery(name = "ReportOptProduct.findBestSeller", query = "SELECT r FROM ReportOptProducts r ORDER BY r.sales DESC") })

public class ReportOptProducts implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    private int idOptionalProduct;
    private int sales;

    public ReportOptProducts() {
    }

}
```

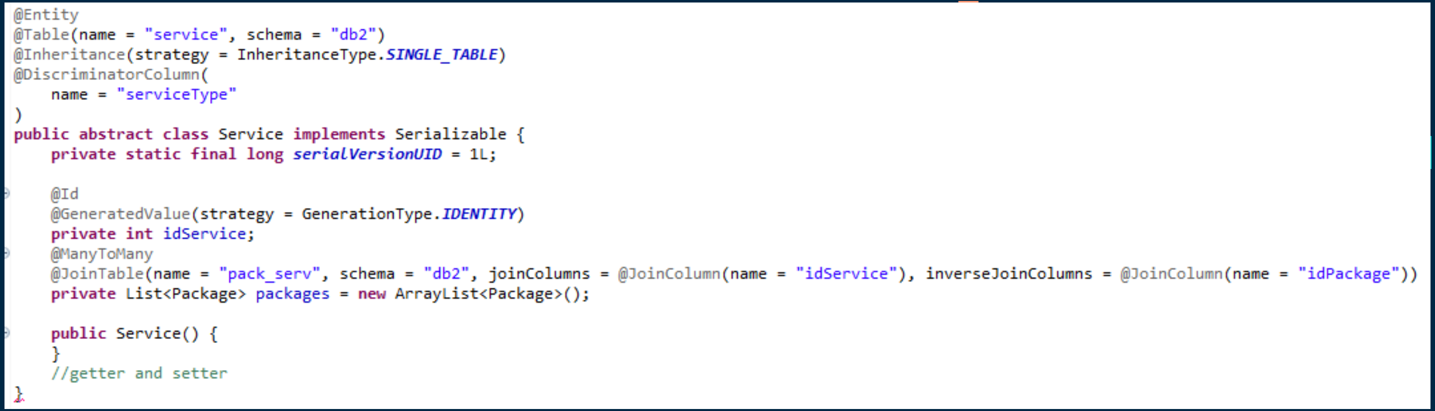
Entity ReportOptProd



```
@Entity
@Table(name = "report_pack", schema = "db2")
@NamedQueries({ @NamedQuery(name = "ReportPack.findAll", query = "SELECT a FROM ReportPack a") })

public class ReportPack implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    private int idPackage;
    private int numPurchase;
    private int totAmount;
    private int totAmountWithoutOpt;
    private float avgOptProducts;

    public ReportPack() {
    }

    //getter and setter

}
```

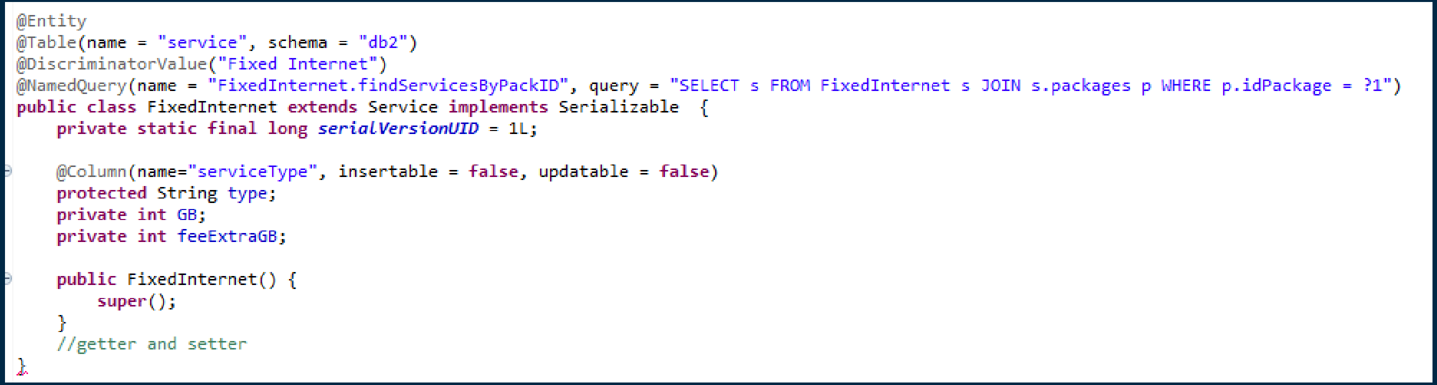
Entity ReportPack

Entity Service

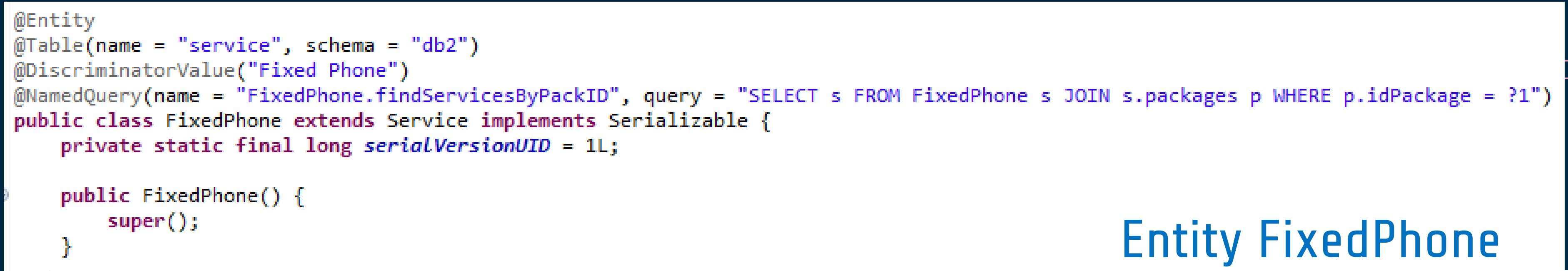


```
@Entity
@Table(name = "service", schema = "db2")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(
    name = "serviceType"
)
public abstract class Service implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idService;
    @ManyToMany
    @JoinTable(name = "pack_serv", schema = "db2", joinColumns = @JoinColumn(name = "idService"), inverseJoinColumns = @JoinColumn(name = "idPackage"))
    private List<Package> packages = new ArrayList<Package>();

    public Service() {
    }
    //getter and setter
}
```

Entity FixedInternet

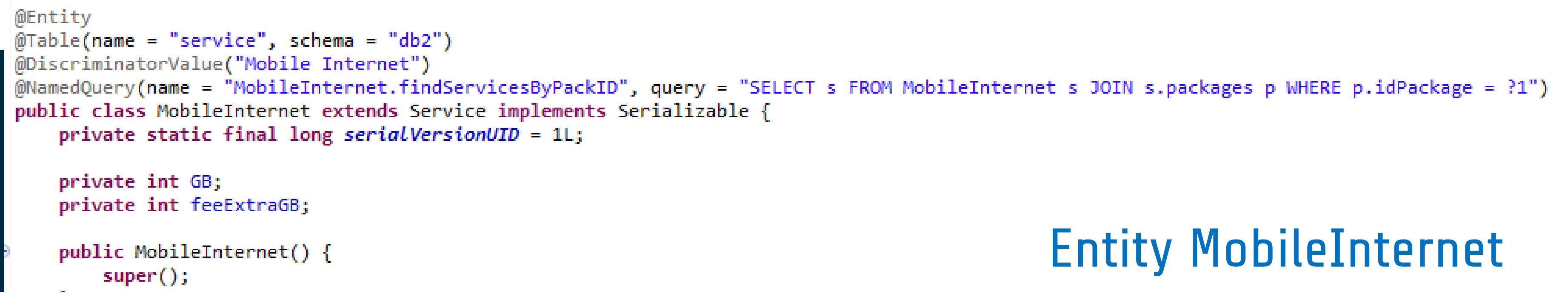


```
@Entity
@Table(name = "service", schema = "db2")
@DiscriminatorValue("Fixed Internet")
@NamedQuery(name = "FixedInternet.findServicesByPackID", query = "SELECT s FROM FixedInternet s JOIN s.packages p WHERE p.idPackage = ?1")
public class FixedInternet extends Service implements Serializable  {
    private static final long serialVersionUID = 1L;

    @Column(name="serviceType", insertable = false, updatable = false)
    protected String type;
    private int GB;
    private int feeExtraGB;

    public FixedInternet() {
        super();
    }
    //getter and setter
}
```



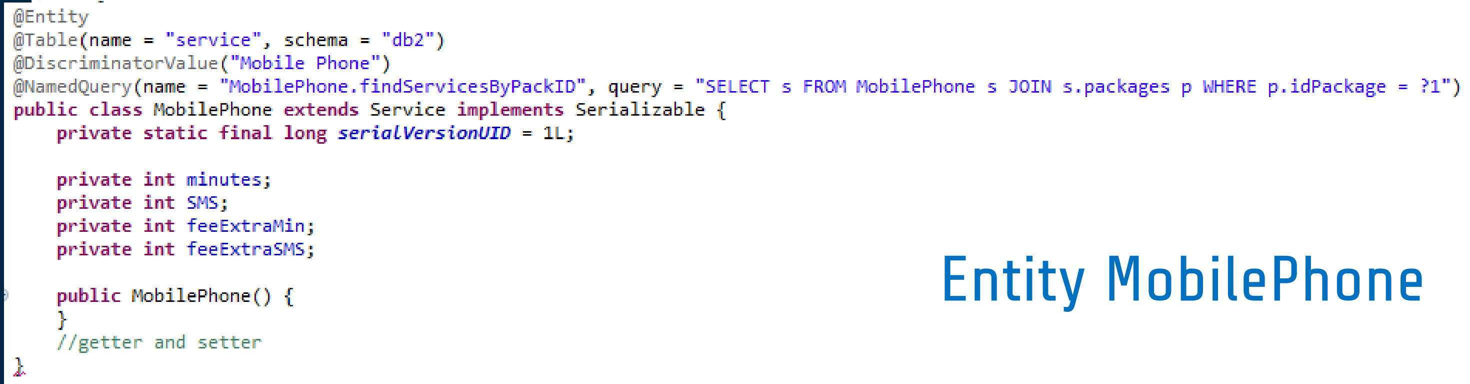
```java
@Entity
@Table(name = "service", schema = "db2")
@DiscriminatorValue("Fixed Phone")
@NamedQuery(name = "FixedPhone.findServicesByPackID", query = "SELECT s FROM FixedPhone s JOIN s.packages p WHERE p.idPackage = ?1")
public class FixedPhone extends Service implements Serializable {
    private static final long serialVersionUID = 1L;

    public FixedPhone() {
        super();
    }

}
```

Entity FixedPhone



```java
@Entity
@Table(name = "service", schema = "db2")
@DiscriminatorValue("Mobile Internet")
@NamedQuery(name = "MobileInternet.findServicesByPackID", query = "SELECT s FROM MobileInternet s JOIN s.packages p WHERE p.idPackage = ?1")
public class MobileInternet extends Service implements Serializable {
    private static final long serialVersionUID = 1L;

    private int GB;
    private int feeExtraGB;

    public MobileInternet() {
        super();
```

Entity MobileInternet

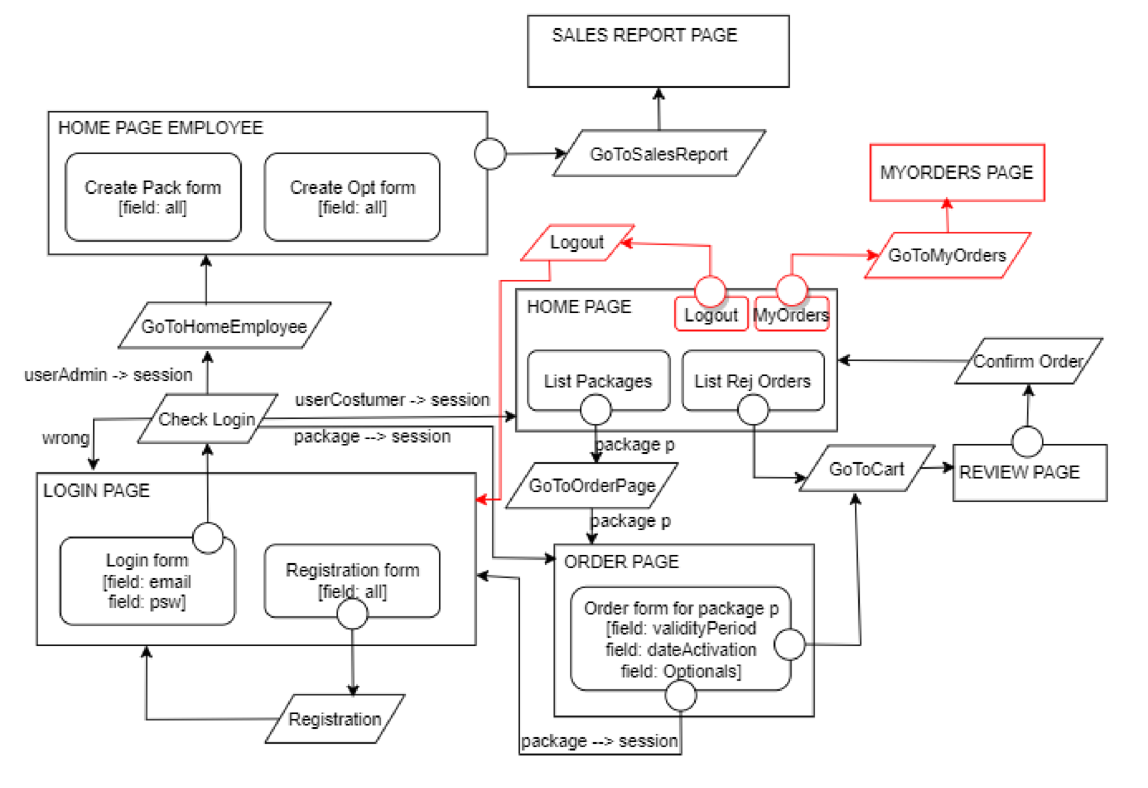


```java
@Entity
@Table(name = "service", schema = "db2")
@DiscriminatorValue("Mobile Phone")
@NamedQuery(name = "MobilePhone.findServicesByPackID", query = "SELECT s FROM MobilePhone s JOIN s.packages p WHERE p.idPackage = ?1")
public class MobilePhone extends Service implements Serializable {
    private static final long serialVersionUID = 1L;

    private int minutes;
    private int SMS;
    private int feeExtraMin;
    private int feeExtraSMS;

    public MobilePhone() {
    }
    //getter and setter
}
```

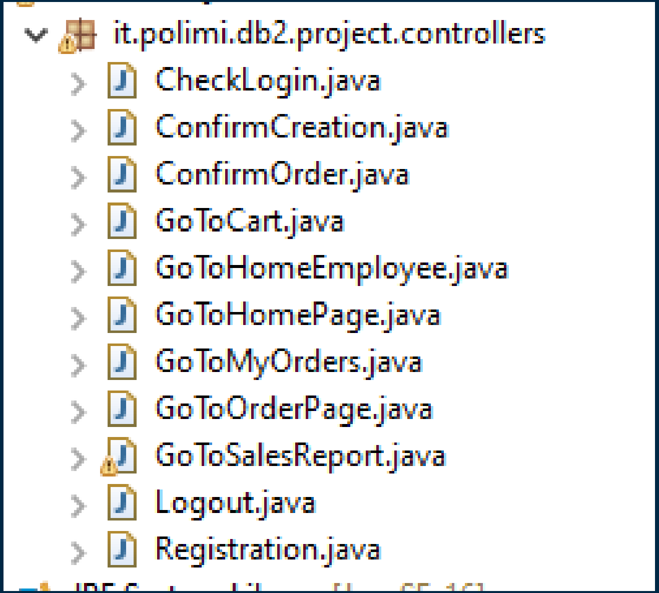
Entity MobilePhone

Interaction diagram

Tutte le pagine hanno la parte di navigazione rossa e un «back to home» che torna alla HomePage, ma per semplicità l'abbiamo ripetuto una sola volta



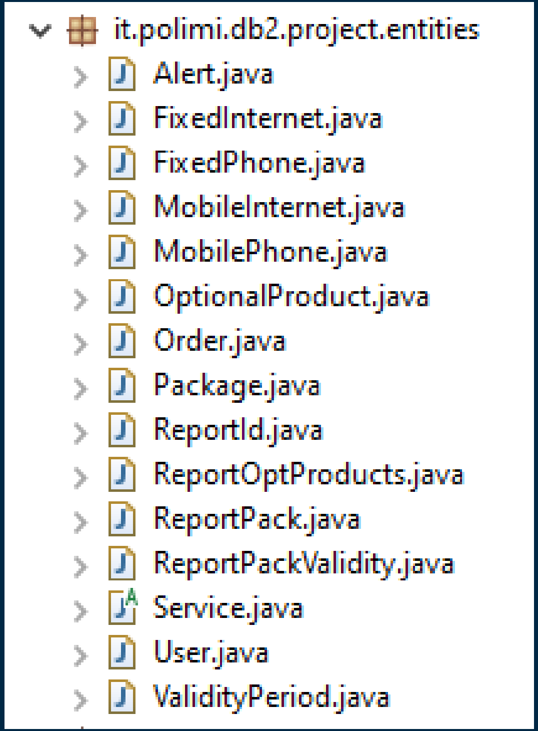
Client components



it.polimi.db2.project.controllers

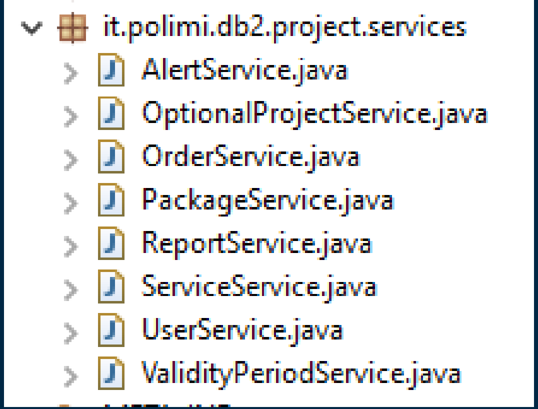
- CheckLogin.java
- ConfirmCreation.java
- ConfirmOrder.java
- GoToCart.java
- GoToHomeEmployee.java
- GoToHomePage.java
- GoToMyOrders.java
- GoToOrderPage.java
- GoToSalesReport.java
- Logout.java
- Registration.java

Backend components



it.polimi.db2.project.entities

- Alert.java
- FixedInternet.java
- FixedPhone.java
- MobileInternet.java
- MobilePhone.java
- OptionalProduct.java
- Order.java
- Package.java
- ReportId.java
- ReportOptProducts.java
- ReportPack.java
- ReportPackValidity.java
- Service.java
- User.java
- ValidityPeriod.java



it.polimi.db2.project.services

- AlertService.java
- OptionalProjectService.java
- OrderService.java
- PackageService.java
- ReportService.java
- ServiceService.java
- UserService.java
- ValidityPeriodService.java

THANKS

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik