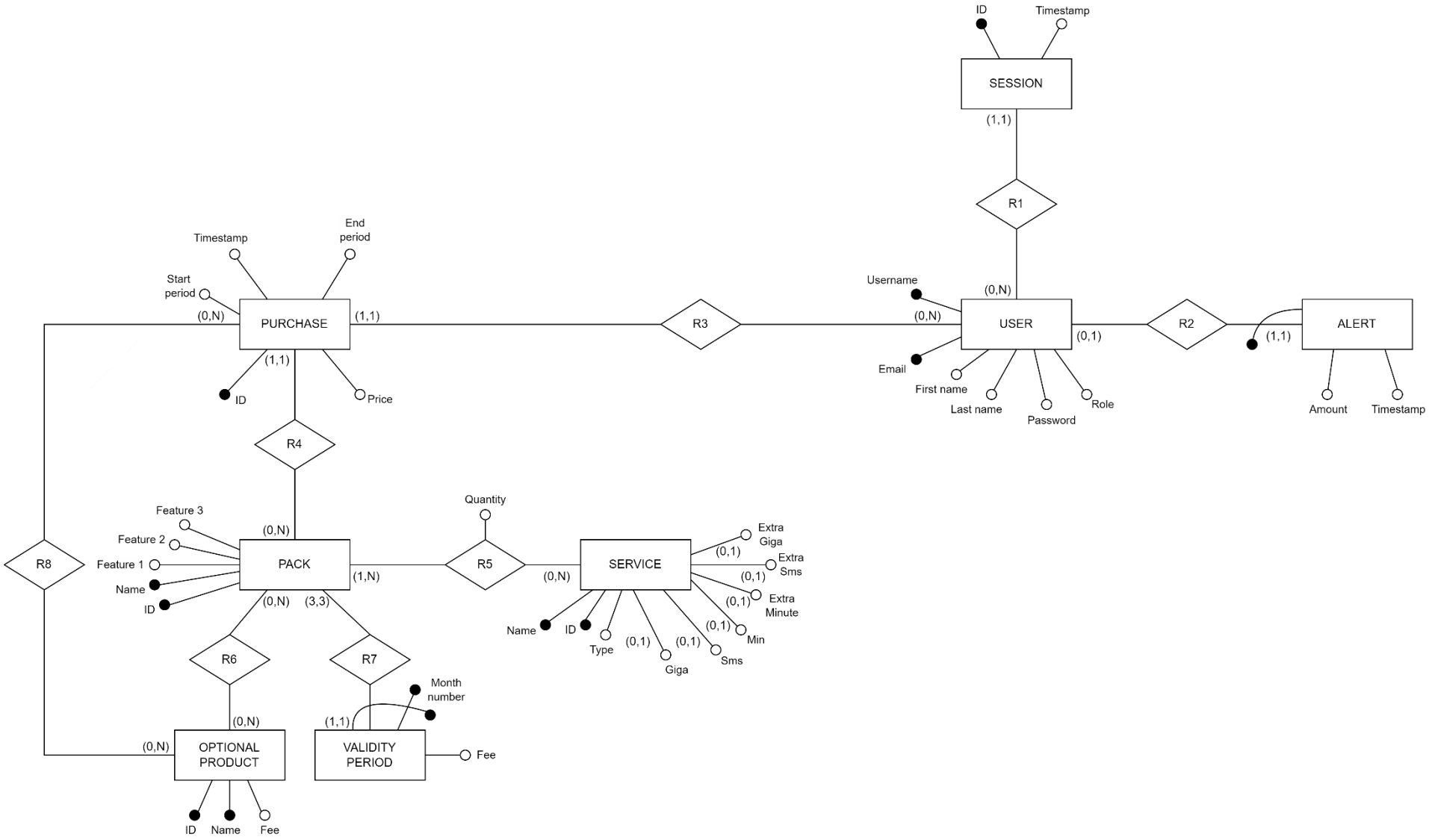
Data bases 2

Telco project documentation

Index

- Conceptual (ER)
- Relational model in SQL
- Sales Report tables
- Triggers
- ORM relationship design
- Entities code
- List of components
- UML sequence diagrams

Entity Relationship



SQL tables

```
TABLE user(
username VARCHAR (20) PRIMARY KEY,
firstname VARCHAR(20) NOT NULL,
lastname VARCHAR(20) NOT NULL,
password VARCHAR(15) NOT NULL,
email VARCHAR(40) UNIQUE NOT NULL,
role ENUM('consumer', 'employee') NOT NULL)

TABLE session(
id SERIAL PRIMARY KEY,
timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
user VARCHAR(20) NOT NULL,
FOREIGN KEY (user) REFERENCES user(username)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

```
TABLE pack(
id SERIAL PRIMARY KEY,
name VARCHAR(30) UNIQUE NOT NULL,
feature1 VARCHAR(30),
feature2 VARCHAR(30),
feature3 VARCHAR(30))

TABLE service(
id SERIAL PRIMARY KEY,
name VARCHAR(30) UNIQUE NOT NULL,
type ENUM('fixedphone', 'mobilephone', 'fixedinternet', 'mobileinternet') NOT NULL,
giga INT CHECK (giga >= -1) COMMENT 'giga = -1 unlimited',
sms INT CHECK (sms >= -1) COMMENT 'sms = -1 unlimited',
min INT CHECK (min >= -1) COMMENT 'min = -1 unlimited',
extragiga FLOAT CHECK (extragiga >= 0) COMMENT 'fee for each extra giga used'
extrasms FLOAT CHECK (extrasms >= 0) COMMENT 'fee for each extra sms used',
extraminute FLOAT CHECK (extraminute >= 0) COMMENT 'fee for each extra min used')

TABLE optionalproduct(
id SERIAL PRIMARY KEY,
name VARCHAR(20) UNIQUE NOT NULL,
fee FLOAT CHECK (fee >= 0) COMMENT 'fee=0 means gratis')
```

```
TABLE packageservice(
service BIGINT UNSIGNED,
pack BIGINT UNSIGNED,
quantity INT CHECK (quantity > 0),
PRIMARY KEY(service, pack),
FOREIGN KEY (service) REFERENCES service(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE)

TABLE packageproduct(
optionalproduct BIGINT UNSIGNED,
pack BIGINT UNSIGNED,
PRIMARY KEY(optionalproduct, pack),
FOREIGN KEY (optionalproduct) REFERENCES optionalproduct(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

```
TABLE purchase(
id SERIAL PRIMARY KEY,
price FLOAT CHECK (price>0),
timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
startperiod DATE NOT NULL,
endperiod DATE NOT NULL,
user VARCHAR(20),
pack BIGINT UNSIGNED,
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (user) REFERENCES user(username)
ON DELETE NO ACTION ON UPDATE CASCADE)

TABLE purchaseproduct(
purchase BIGINT UNSIGNED,
optionalproduct BIGINT UNSIGNED,
PRIMARY KEY(optionalproduct, purchase),
FOREIGN KEY (optionalproduct) REFERENCES optionalproduct(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (purchase) REFERENCES purchase(id)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

```
TABLE validityperiod(
monthnumber ENUM('12', '24', '36’),
pack BIGINT UNSIGNED,
fee FLOAT CHECK (fee > 0),
PRIMARY KEY(monthnumber, pack),
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE)

TABLE alert(
user VARCHAR(20) PRIMARY KEY,
email VARCHAR(40) UNIQUE NOT NULL,
amount FLOAT CHECK (amount > 0),
timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
FOREIGN KEY (user) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE NO ACTION,
FOREIGN KEY (email) REFERENCES user(email)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

Sales Report tables

```
CREATE TABLE userSalesReport(
user VARCHAR (20) PRIMARY KEY,
rejectedpurchases INT CHECK (rejectedpurchases >= 0),
solvent BOOL NOT NULL,
FOREIGN KEY (user) REFERENCES user(username)
ON DELETE CASCADE ON UPDATE NO ACTION)

CREATE TABLE packSalesReport(
pack BIGINT UNSIGNED PRIMARY KEY,
purchasesop INT CHECK (purchasesop >=0),
purchasesnoop INT CHECK (purchasesnoop >=0),
averageproduct FLOAT CHECK (averageproduct >=0),
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE)

CREATE TABLE optionalproductSalesReport(
optionalproduct BIGINT UNSIGNED PRIMARY KEY,
amountsold INT NOT NULL CHECK (amountsold >= 0),
FOREIGN KEY (optionalproduct) REFERENCES optionalproduct(id)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

```
CREATE TABLE purchaseSalesReport(
purchase BIGINT UNSIGNED PRIMARY KEY,
rejected INT CHECK (rejected >= 0),
optionalproduct INT CHECK (optionalproduct >= 0),
price FLOAT CHECK (price>0),
user VARCHAR(20), pack BIGINT UNSIGNED,
FOREIGN KEY (purchase) REFERENCES purchase(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (pack) REFERENCES pack(id)
ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (user) REFERENCES user(username)
ON DELETE NO ACTION ON UPDATE CASCADE)

CREATE TABLE validityperiodSalesReport(
monthnumber ENUM('12', '24', '36’),
pack BIGINT UNSIGNED,
quantity INT,
PRIMARY KEY(monthnumber, pack),
FOREIGN KEY (monthnumber, pack) REFERENCES validityperiod(monthnumber,pack)
ON DELETE NO ACTION ON UPDATE CASCADE)
```

Triggers design & code

- Event: after insertion of a new user
- Condition: none
- Action: a new tuple having rejectedpurchases = 0 and solvent = true is added to userSalesReport

```
CREATE TRIGGER insert_userSalesReport
AFTER INSERT ON user
FOR EACH ROW
BEGIN
INSERT INTO userSalesReport(user, rejectedpurchases, solvent)
VALUES(NEW.username, 0, true);
END;
```

- Event: after insertion of a new pack
- Condition: none
- Action: a new tuple having purchasesop = 0, purchasesnoop = 0 and averageproduct = 0 is added to packSalesReport

```
CREATE TRIGGER insert_packSalesReport
AFTER INSERT ON pack
FOR EACH ROW
BEGIN
INSERT INTO packSalesReport(pack, purchasesop, purchasesnoop, averageproduct)
VALUES(NEW.id, 0, 0, 0);
END;
```

- Event: after insertion of a new optionalproduct
- Condition: none
- Action: a new tuple having amountsold = 0 is added to optionalproductSalesReport

```
CREATE TRIGGER insert_optionalproductSalesReport
AFTER INSERT ON optionalproduct
FOR EACH ROW
BEGIN
INSERT INTO optionalproductSalesReport(optionalproduct, amountsold)
VALUES(NEW.id, 0);
END;
```

- Event: after insertion of a new validityperiod
- Condition: none
- Action: a new tuple having quantity = 0 is added to validityperiodSalesReport

```
CREATE TRIGGER insert_validityperiodSalesReport
AFTER INSERT ON validityperiod
FOR EACH ROW
BEGIN
INSERT INTO validityperiodSalesReport(monthnumber, pack, quantity)
VALUES(NEW.monthnumber, NEW.pack, 0);
END;
```

- Event: after insertion of a new purchase
- Condition: none
- Action: increase by one both purchasesnoop of packSalesReport and quantity of the validityperiodSalesReport

```
CREATE TRIGGER number_of_purchasesnoop

AFTER INSERT ON purchase

FOR EACH ROW

BEGIN

UPDATE packSalesReport

SET purchasesnoop = purchasesnoop+1

WHERE pack = NEW.pack;

UPDATE validityperiodSalesReport

SET quantity = quantity+1

WHERE CAST(CAST(monthnumber AS CHAR) AS SIGNED) = TIMESTAMPDIFF(month, NEW.startperiod, NEW.endperiod) AND pack = NEW.pack;

END;
```

- Event: after insertion of a new tuple purchaseproduct
- Condition: purchaseproduct table has only one row where purchase ID is equal to the one inserted
- Action: increase purchasesop and decrease purchasenoop by one

```
CREATE TRIGGER number_of_purchasesop

AFTER INSERT ON purchaseproduct

FOR EACH ROW

BEGIN

IF (SELECT COUNT(*) FROM purchaseproduct WHERE purchase = NEW.purchase) = 1
THEN

UPDATE packSalesReport

SET purchasesop = purchasesop+1, purchasesnoop = purchasesnoop-1

WHERE pack = (SELECT pack FROM purchase WHERE id = NEW.purchase);

END IF;

END;
```

- Event: after insertion of a new purchaseSalesReport
- Condition: the purchase inserted, having rejected>0, is rejected
- Action: the attribute solvent of userSalesReport is set to false, the number of rejectedpurchases is increased by one

```
CREATE TRIGGER insert_rejectedpurchase
AFTER INSERT ON purchaseSalesReport
FOR EACH ROW
IF NEW.rejected > 0 THEN
BEGIN
UPDATE userSalesReport
SET solvent = false,
rejectedpurchases = rejectedpurchases + 1
WHERE user = NEW.user;
END;
END IF;
```

- Event: after update of a purchaseSalesReport
- Condition: the updated purchaseSalesReport has an increase in the value of the rejected attribute
- Action: the number of rejectedpurchases is increased by the difference NEW.rejected - OLD.rejected

```
CREATE TRIGGER update_rejectedpurchase
AFTER UPDATE ON purchaseSalesReport
FOR EACH ROW
IF NEW.rejected > OLD.rejected THEN
BEGIN
UPDATE userSalesReport
SET rejectedpurchases = rejectedpurchases + NEW.rejected - OLD. rejected

WHERE user = NEW.user;
END;
END IF;
```

- Event: after update of a purchaseSalesReport
- Condition: the updated purchaseSalesReport has the rejected attribute equal to zero, NEW.rejected = 0, while previously it had OLD.rejected>0 and this was the only row of purchaseSalesReport having rejected != 0
- Action: the user attribute solvent is set to true

```
CREATE TRIGGER accepted_purchase

AFTER UPDATE ON purchaseSalesReport

FOR EACH ROW

IF NEW.rejected = 0 AND OLD.rejected > 0 AND (SELECT COUNT(*) FROM
purchaseSalesReport WHERE rejected != 0 AND user = NEW.user) = 0 THEN

BEGIN

UPDATE userSalesReport

SET solvent = true

WHERE user = NEW.user;

END;

END IF;
```

- Event: after insertion of a new tuple in purchaseproduct
- Condition: none
- Action: increase amountsold value by one

```
CREATE TRIGGER amountsold
AFTER INSERT ON purchaseproduct
FOR EACH ROW
BEGIN
UPDATE optionalproductSalesReport
SET amountsold = amountsold+1
WHERE optionalproduct = NEW.optionalproduct;
END;
```

- Event: after update of a userSalesReport
- Condition: the updated userSalesReport has a number of rejectedpurchases different from zero and multiple of 3
- Action: a new alert for the user is created

```
CREATE TRIGGER create_alert

AFTER UPDATE ON userSalesReport

FOR EACH ROW

IF NEW.rejectedpurchases % 3 = 0 AND NEW.rejectedpurchases !=  0 AND
NEW.solvent = 0  THEN

BEGIN

DECLARE alert_price FLOAT;

DECLARE user_email varchar(40);

SELECT SUM(price) FROM purchasesalesreport WHERE user = NEW.user AND rejected > 0
INTO alert_price;

SELECT email FROM user WHERE username = NEW.user INTO user_email;
```

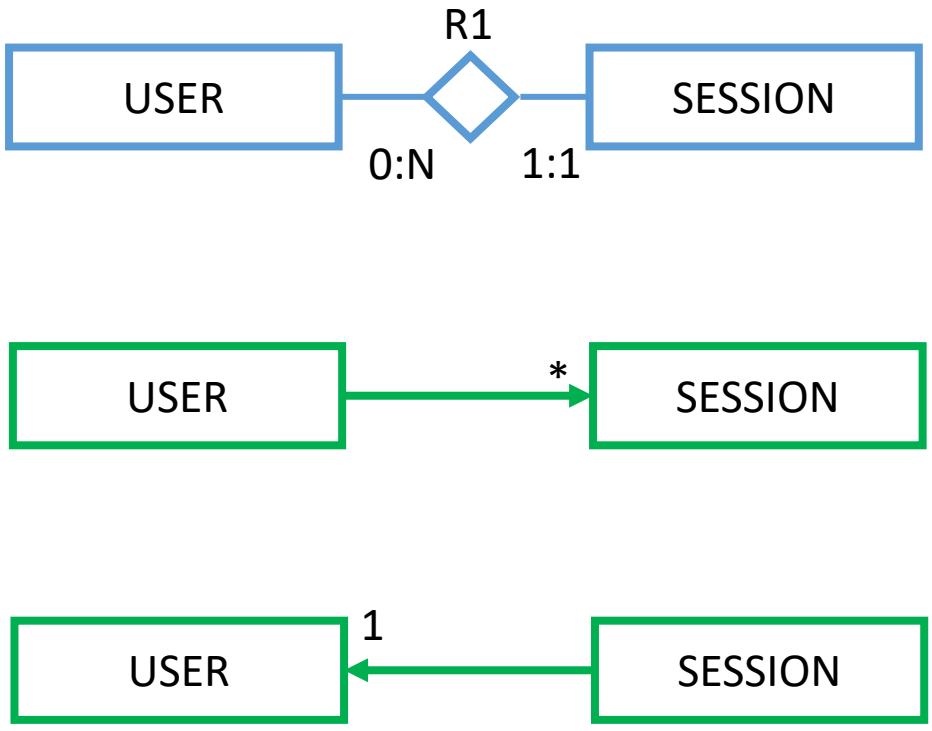
```
IF (SELECT COUNT(*) FROM alert WHERE user = NEW.user) = 0 THEN
INSERT INTO alert(user, email, amount)
VALUES(NEW.user, user_email, alert_price);
ELSE
UPDATE alert
SET timestamp = now(), amount = alert_price
WHERE user = NEW.user;
END IF;
END;
END IF;
```

- Event: after insertion of a new purchaseSalesReport
- Condition: none
- Action: compute and set to averageproduct the value of the average number of optional products sold together with each service package

```
CREATE TRIGGER average_products
AFTER INSERT ON purchaseSalesReport
FOR EACH ROW
BEGIN
DECLARE total_pack integer;
DECLARE total_product integer;
SELECT COUNT(*) FROM purchase WHERE pack = NEW.pack INTO total_pack;
SELECT SUM(optionalproduct) FROM purchaseSalesReport WHERE pack = NEW.pack
INTO total_product;
UPDATE packSalesReport
SET averageproduct = total_product/total_pack
WHERE pack = NEW.pack;
END;
```

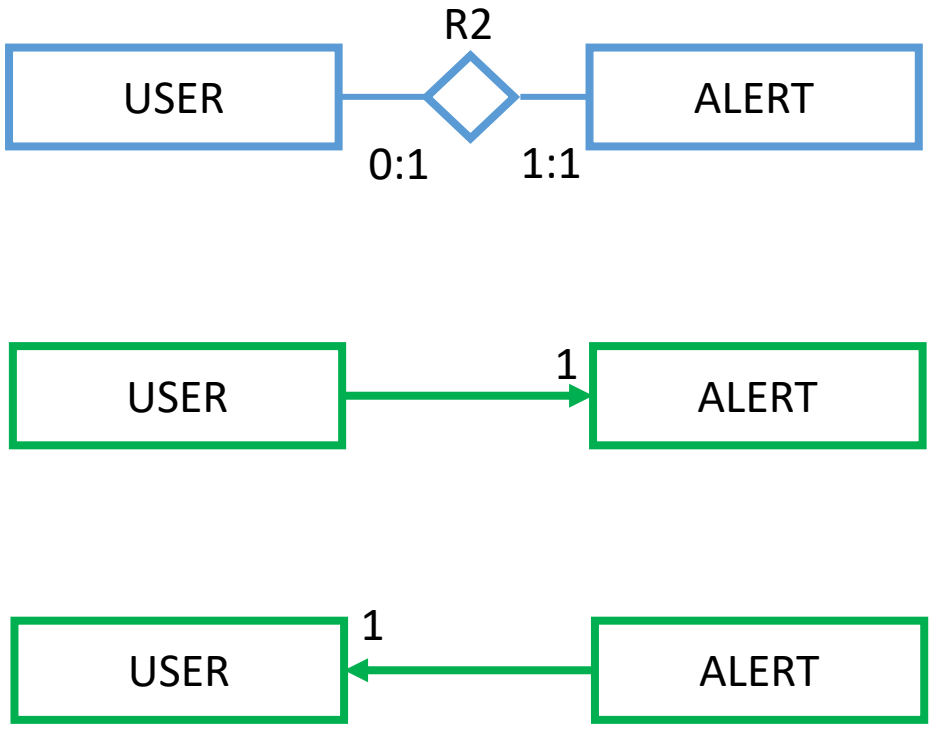
ORM design

Relationship “R1”



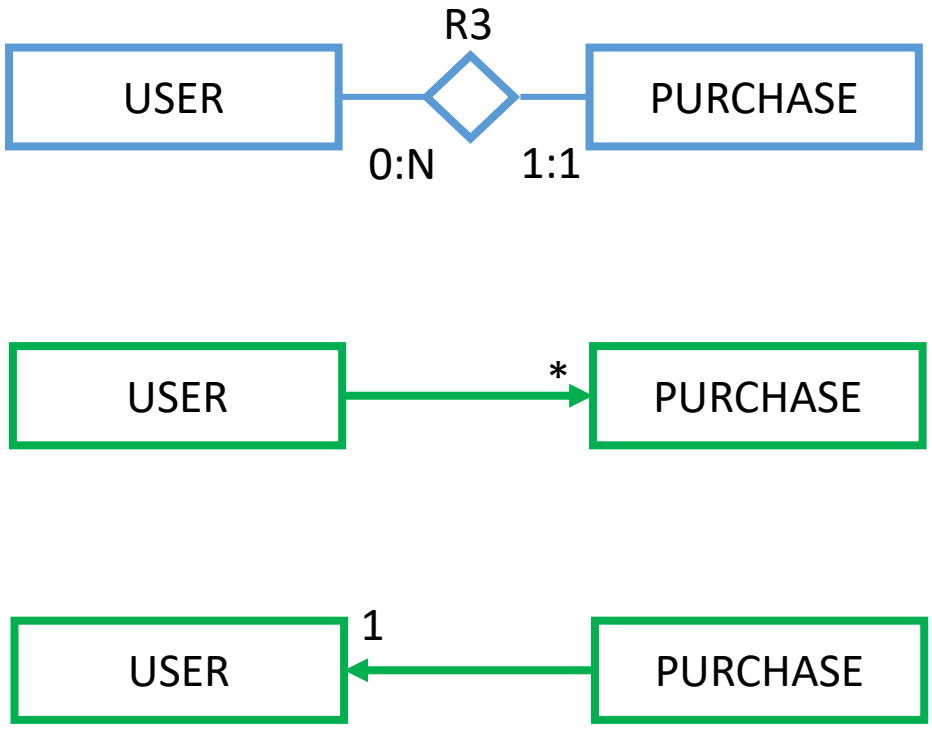
- USER → SESSION @OneToMany is necessary to get the sessions associated to the user
  - Owner = session
  - Cascade Type = Persist
- SESSION → USER @ManyToOne is mapped for simplicity as well

Relationship “R2”



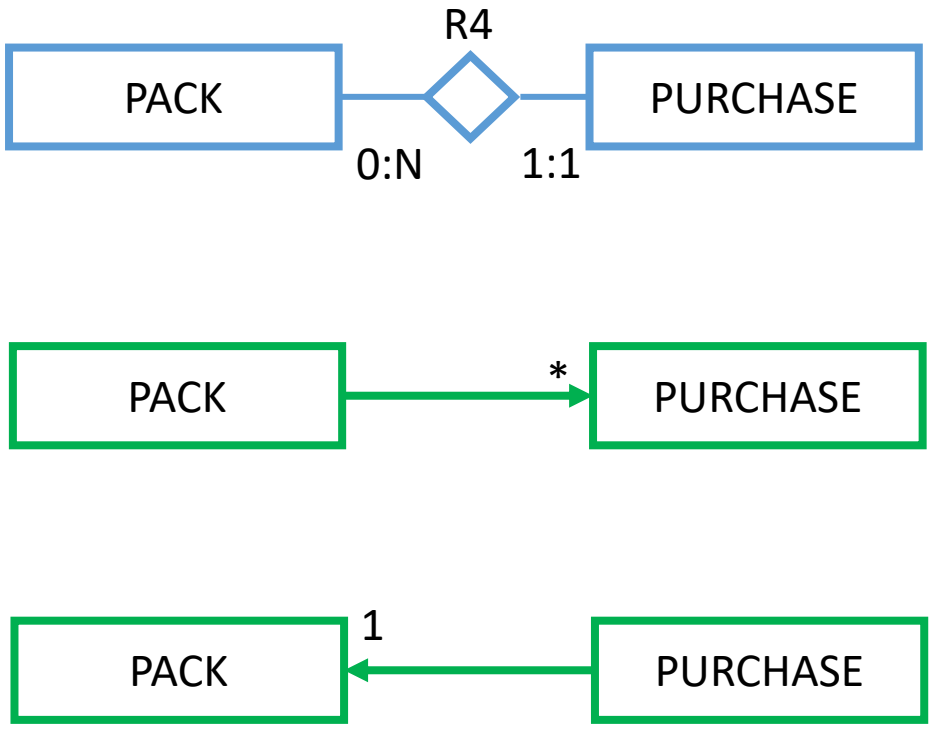
- USER → ALERT
  @OneToOne is necessary to get the alerts associated to the user
  - Owner = either alert or user
  - Cascade Type = Persist
- ALERT → USER
  @OneToOne is mapped for simplicity as well

Relationship “R3”



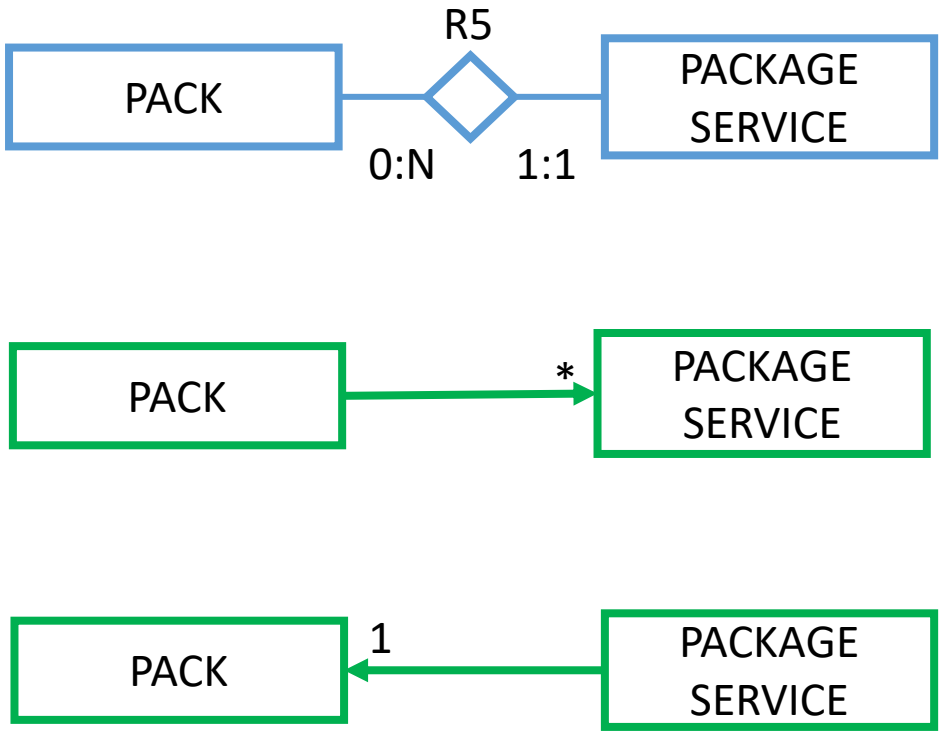
- USER → PURCHASE @OneToMany is necessary to get the purchases associated to the user
  - Owner = purchase
  - Cascade Type = Persist
- PURCHASE → USER @ManyToOne is mapped for simplicity as well

Relationship “R4”



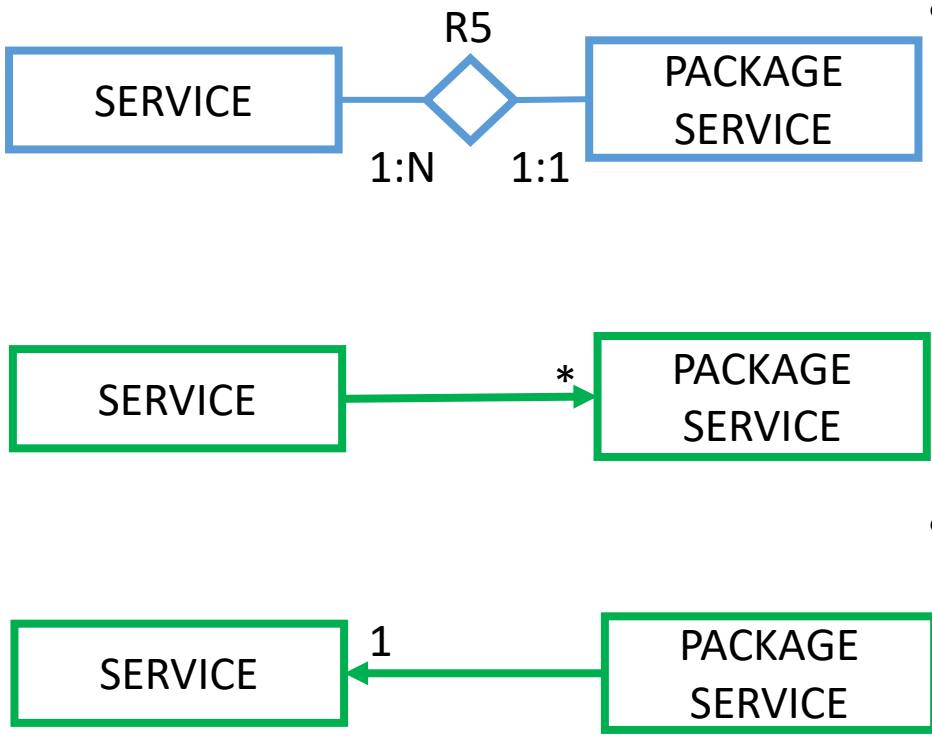
- PACK → PURCHASE @OneToMany is necessary to get the purchases associated to the pack
  - Owner = purchase
  - Cascade Type = Persist
- PURCHASE → PACK @ManyToOne is mapped for simplicity as well

Relationship “R5”



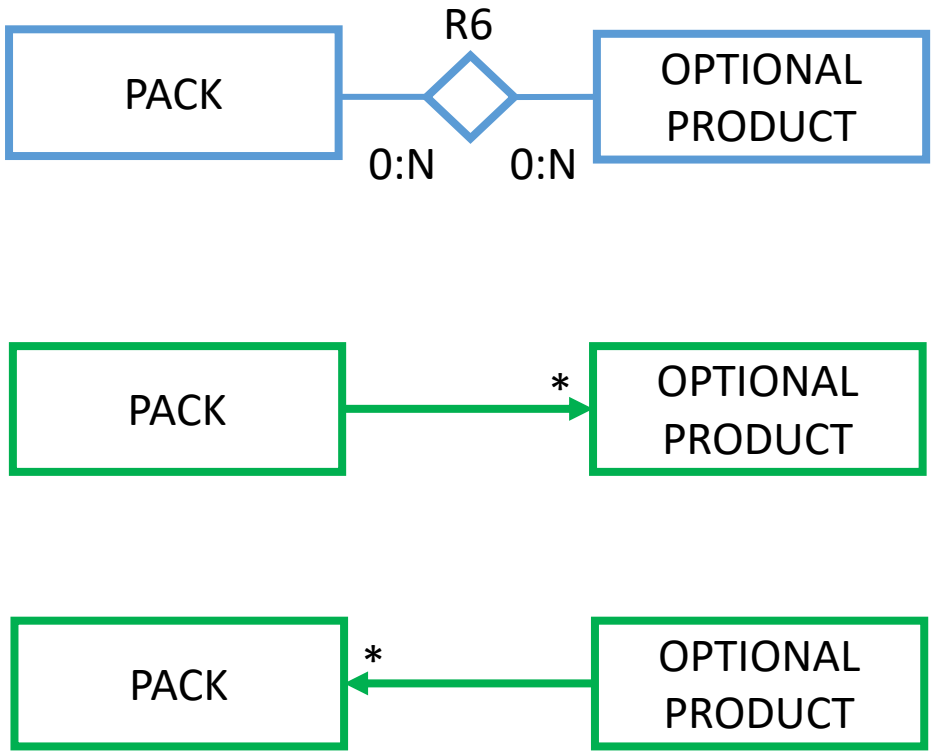
- PACK → PACKAGE SERVICE @OneToMany is necessary to show the package services assigned to each pack
  - Owner = package service
  - Cascade Type = Persist
- PACKAGE SERVICE →PACK @ManyToOne is mapped for simplicity as well

Relationship “R5”



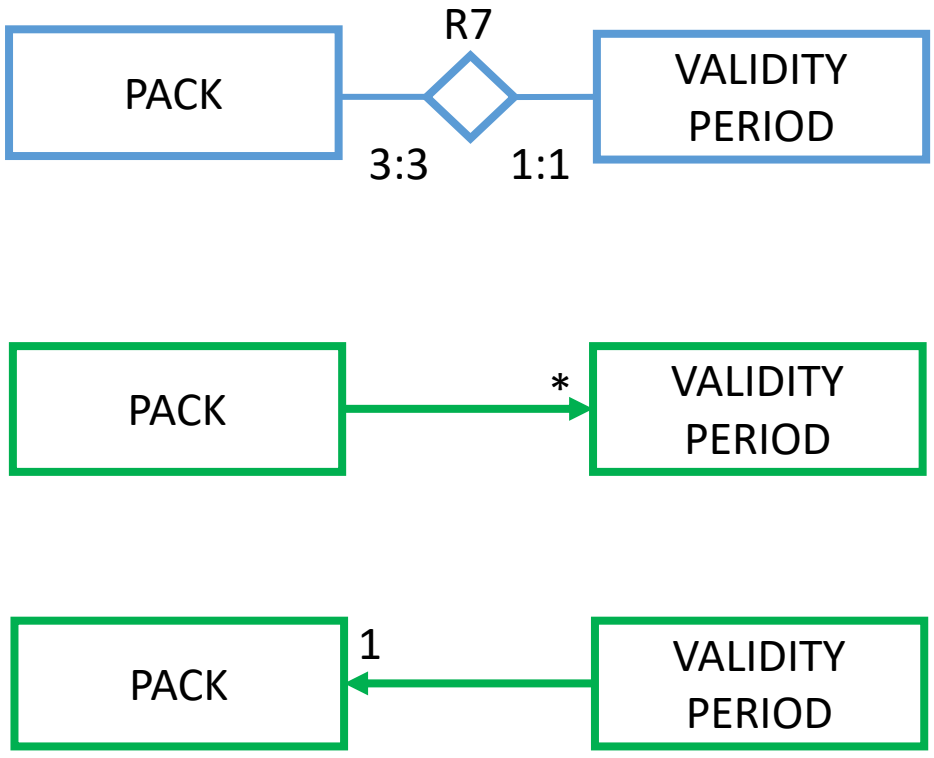
- SERVICE → PACKAGE SERVICE @OneToMany is necessary to show the package services assigned to each service
  - Owner = package service
  - Cascade Type = Persist
- PACKAGE SERVICE →SERVICE @ManyToOne is mapped for simplicity as well

Relationship “R6”



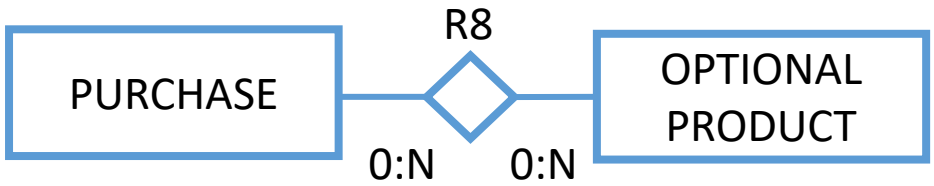
- PACK → OP. PRODUCT @ManyToMany is necessary to show the optional products assigned to each pack
  - Owner = either pack or optional product
  - Cascade Type = Persist
- OP. PRODUCT → PACK @ManyToMany is mapped for simplicity as well

Relationship “R7”



- PACK → V. PERIOD @OneToMany is necessary to get the validity periods associated to each pack
  - Owner = validity period
  - Cascade Type = Persist
- V. PERIOD → PACK @ManyToOne is mapped for simplicity as well

Relationship “R8”



- PURCHASE → OP. PRODUCT @ManyToMany is necessary to show the products associated to each purchase
  - Owner = either purchase or optional product
  - Cascade Type = Persist
- OP. PRODUCT → PURCHASE @ManyToMany is mapped for simplicity as well

Entity Alert

```
@Entity

@Table(name="alert")

@NamedQuery(name="Alert.findAll", query="SELECT a FROM Alert a")

public class Alert implements Serializable {

            private static final long serialVersionUID = 1L;

            @Id

            @Column(unique=true, nullable=false, insertable=false, updatable=false, length=20)

            private String user;

            @OneToOne(cascade={CascadeType.PERSIST})

            @JoinColumn(name="user", referencedColumnName="username", nullable=false, insertable=false,
            updatable=false)

            private User userBean;

            @Column(insertable=false, updatable=false, length=40)

            private float amount;

            @Column(nullable=false, insertable=false, updatable=false, length=40)

            private String email;

            @Column(nullable=false, insertable=false, updatable=false)

            private Timestamp timestamp;
```

Entity Optional product

```
@Entity
@Table(name="optionalproduct")
@NamedQuery(name="Optionalproduct.findAll", query="SELECT o FROM Optionalproduct o")
public class Optionalproduct implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @GeneratedValue(strategy=GenerationType.IDENTITY)
            @Column(unique=true, nullable=false)
            private long id;
            private float fee;
            @Column(nullable=false, length=20)
            private String name;
            @ManyToMany(mappedBy="optionalproducts", cascade={CascadeType.PERSIST})
            private List<Pack> packs;
            @ManyToMany(mappedBy="optionalproducts", cascade={CascadeType.PERSIST})
            private List<Purchase> purchases;
            @OneToOne(mappedBy="optionalproductBean", cascade={CascadeType.PERSIST})
            private Optionalproductsalesreport optionalproductsalesreport;
```

Entity Pack

```
@Entity
@Table(name="pack")
@NamedQuery(name="Pack.findAll", query="SELECT p FROM Pack p")
public class Pack implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @GeneratedValue(strategy=GenerationType.IDENTITY)
            @Column(unique=true, nullable=false)
            private long id;
            @Column(length=30)
            private String feature1;
            @Column(length=30)
            private String feature2;
            @Column(length=30)
            private String feature3;
            @Column(nullable=false, length=30)
            private String name;
            @OneToMany(mappedBy="packBean", cascade={CascadeType.PERSIST})
            private List<Packageservice> packageservices;
            @OneToMany(mappedBy="packBean", cascade={CascadeType.PERSIST})
            private List<Purchase> purchases;
```

```
@ManyToMany(cascade={CascadeType.PERSIST})
@JoinTable(name="packageproduct"
, joinColumns={@JoinColumn(name="pack", nullable=false)}
, inverseJoinColumns={@JoinColumn(name="optionalproduct", nullable=false)})
private List<Optionalproduct> optionalproducts;
@OneToMany(mappedBy="packBean", cascade={CascadeType.PERSIST})
private List<Purchasesalesreport> purchasesalesreports;
@OneToMany(mappedBy="packBean", cascade={CascadeType.PERSIST})
private List<Validityperiod> validityperiods;
@OneToOne(mappedBy="packBean", cascade={CascadeType.PERSIST})
private Packsalesreport packsalesreport;
```

Entity Package service

```
@Entity
@Table(name="packageservice")
@NamedQuery(name="Packageservice.findAll", query="SELECT p FROM Packageservice p")
public class Packageservice implements Serializable {
            private static final long serialVersionUID = 1L;
            @EmbeddedId
            private PackageservicePK id;
            private int quantity;
            @ManyToOne(cascade={CascadeType.PERSIST})
            @JoinColumn(name="pack", nullable=false, insertable=true, updatable=true)
            private Pack packBean;
            @ManyToOne(cascade={CascadeType.PERSIST})
            @JoinColumn(name="service", nullable=false, insertable=true, updatable=true)
            private Service serviceBean;
```

Entity Package service PK

```
@Embeddable
public class PackageservicePK implements Serializable {
            private static final long serialVersionUID = 1L;
            @Column(insertable=true, updatable=true, unique=true, nullable=false)
            private long service;
            @Column(insertable=true, updatable=true, unique=true, nullable=false)
            private long pack;
```

Entity Purchase

```
@Entity
@Table(name="purchase")
@NamedQuery(name="Purchase.findAll", query="SELECT p FROM Purchase p")
public class Purchase implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @GeneratedValue(strategy=GenerationType.IDENTITY)
            @Column(unique=true, nullable=false)
            private long id;
            @Temporal(TemporalType.DATE)
            @Column(nullable=false)
            private Date endperiod;
            private float price;
```

```
@Temporal(TemporalType.DATE)
@Column(nullable=false)
private Date startperiod;
@Column(nullable=false)
private Timestamp timestamp;
@ManyToMany(cascade={CascadeType.PERSIST})
@JoinTable(name="purchaseproduct"
, joinColumns={@JoinColumn(name="purchase", nullable=false)}
, inverseJoinColumns={@JoinColumn(name="optionalproduct", nullable=false)})
private List<Optionalproduct> optionalproducts;
@ManyToOne(cascade={CascadeType.PERSIST})
@JoinColumn(name="pack")
private Pack packBean;
@ManyToOne(cascade={CascadeType.PERSIST})
@JoinColumn(name="user", referencedColumnName="username")
private User userBean;
@OneToOne(mappedBy="purchaseBean", cascade={CascadeType.PERSIST})
private Purchasesalesreport purchasesalesreport;
```

Entity Service

```
@Entity
@Table(name="service")
@NamedQuery(name="Service.findAll", query="SELECT s FROM Service s")
@NamedQuery(name="Service.findAllType", query = "SELECT s FROM Service s WHERE s.type = ?1")
public class Service implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @GeneratedValue(strategy=GenerationType.IDENTITY)
            @Column(unique=true, nullable=false)
            private long id;
            private float extragiga;
            private float extraminute;
            private float extrasms;
            private int giga;
            private int min;
            @Column(nullable=false, length=30)
            private String name;
            private int sms;
            @Column(nullable=false, length=15)
            private String type;
            @OneToMany(mappedBy="serviceBean", cascade={CascadeType.PERSIST})
            private List<Packageservice> packageservices;
```

Entity Validity period PK

```
@Embeddable
public class ValidityperiodPK implements Serializable {
            private static final long serialVersionUID = 1L;
            @Column(unique=true, nullable=false, length=2)
            private String monthnumber;
            @Column(insertable=true, updatable=true, unique=true, nullable=false)
            private long pack;
```

Entity Session

```
@Entity
@Table(name="session")
@NamedQuery(name="Session.findAll", query="SELECT s FROM Session s")
public class Session implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @GeneratedValue(strategy=GenerationType.IDENTITY)
            @Column(unique=true, nullable=false)
            private long id;
            @Column(nullable=false)
            private Timestamp timestamp;
            @ManyToOne(cascade={CascadeType.PERSIST})
            @JoinColumn(name="user", referencedColumnName="username", nullable=false)
            private User userBean;
```

Entity User

```
@Entity
@Table(name="user")
@NamedQuery(name="User.findAll", query="SELECT u FROM User u")
public class User implements Serializable {
            private static final long serialVersionUID = 1L;
            @Id
            @Column(unique=true, nullable=false)
            private String username;
            @Column(nullable=false, length=40)
            private String email;
            @Column(nullable=false, length=20)
            private String firstname;
            @Column(nullable=false, length=20)
            private String lastname;
            @Column(nullable=false, length=15)
            private String password;
            @Column(nullable=false, length=10)
            private String role;
            @OneToMany(mappedBy="userBean", cascade={CascadeType.PERSIST})
            private List<Purchase> purchases;
            @OneToMany(mappedBy="userBean", cascade={CascadeType.PERSIST})
            private List<Session> sessions;
```

```
@OneToOne(mappedBy="userBean", cascade={CascadeType.PERSIST})
private Alert alerts;
@OneToOne(mappedBy="userBean", cascade={CascadeType.PERSIST})
private Usersalesreport usersalesreport;
@OneToMany(mappedBy="userBean", cascade={CascadeType.PERSIST})
private List<Purchasesalesreport> purchasesalesreports;
```

Entity Validity period

```
@Entity
@Table(name="validityperiod")
@NamedQuery(name="Validityperiod.findAll", query="SELECT v FROM Validityperiod v")
public class Validityperiod implements Serializable {
        private static final long serialVersionUID = 1L;
        @EmbeddedId
        private ValidityperiodPK id;
        private float fee;
        @ManyToOne(cascade={CascadeType.PERSIST})
        @JoinColumn(name="pack", nullable=false, insertable=true, updatable=true)
        private Pack packBean;
        @OneToOne(mappedBy="validityperiod", cascade={CascadeType.PERSIST})
        private Validityperiodsalesreport validityperiodsalesreport;
```

Entity Optionalproductsalesreport

```
@Entity
@Table(name="optionalproductsalesreport")
@NamedQuery(name="Optionalproductsalesreport.findAll", query="SELECT o FROM Optionalproductsalesreport o")
public class Optionalproductsalesreport implements Serializable {
		private static final long serialVersionUID = 1L;
		@Id
		@Column(unique=true, nullable=false, insertable=false, updatable=false)
		private long optionalproduct;
		@Column(unique=false, nullable=false, insertable=false, updatable=false)
		private int amountsold;
		@OneToOne(cascade={CascadeType.PERSIST})
		@JoinColumn(name="optionalproduct", nullable=false, insertable=false, updatable=false)
		private Optionalproduct optionalproductBean;
```

Entity Packsalesreport

```
@Entity
@Table(name="packsalesreport")
@NamedQuery(name="Packsalesreport.findAll", query="SELECT p FROM Packsalesreport p")
public class Packsalesreport implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@Column(unique=true, nullable=false, insertable=false, updatable=false)
	private long pack;
	@Column(unique=false, nullable=false, insertable=false, updatable=false)
	private float averageproduct;
	@Column(unique=false, nullable=false, insertable=false, updatable=false)
	private int purchasesnoop;
	@Column(unique=false, nullable=false, insertable=false, updatable=false)
	private int purchasesop;
	@OneToOne(cascade={CascadeType.PERSIST})
	@JoinColumn(name="pack", nullable=false, insertable=false, updatable=false)
	private Pack packBean;
```

Entity Purchasesalesreport

```
@Entity
@Table(name="purchasesalesreport")
@NamedQuery(name="Purchasesalesreport.findAll", query="SELECT p FROM Purchasesalesreport p")
public class Purchasesalesreport implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@Column(unique=true, nullable=false)
	private long purchase;
	private int optionalproduct;
	private int rejected;
	private float price;
	@OneToOne(cascade={CascadeType.PERSIST})
	@JoinColumn(name="purchase", referencedColumnName="id")
	private Purchase purchaseBean;
	@ManyToOne(cascade={CascadeType.PERSIST})
	@JoinColumn(name="user", referencedColumnName="username")
	private User userBean;
	@ManyToOne(cascade={CascadeType.PERSIST})
	@JoinColumn(name="pack")
	private Pack packBean;
```

Entity Usersalesreport

```
@Entity
@Table(name="usersalesreport")
@NamedQuery(name="Usersalesreport.findAll", query="SELECT u FROM Usersalesreport u")
public class Usersalesreport implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@Column(unique=true, nullable=false, insertable=false, updatable=false, length=20)
	private String user;
	@Column(unique=false, nullable=false, insertable=false, updatable=false)
	private int rejectedpurchases;
	@Column(unique=false, nullable=false, insertable=false, updatable=false)
	private boolean solvent;
	@OneToOne(cascade={CascadeType.PERSIST})
	@JoinColumn(name="user", referencedColumnName="username", nullable=false, insertable=false, updatable=false)
	private User userBean;
```

Entity Validityperiodsalesreport

```
@Entity
@Table(name="validityperiodsalesreport")
@NamedQuery(name="Validityperiodsalesreport.findAll", query="SELECT v FROM Validityperiodsalesreport v")
public class Validityperiodsalesreport implements Serializable {
            private static final long serialVersionUID = 1L;
            @EmbeddedId
            private ValidityperiodsalesreportPK id;
            @Column(unique=false, nullable=false, insertable=false, updatable=false)
            private int quantity;
            @OneToOne(cascade={CascadeType.PERSIST})
            @JoinColumns({
                        @JoinColumn(name="monthnumber", referencedColumnName="monthnumber", nullable=false,
                        insertable=true, updatable=true),
                        @JoinColumn(name="pack", referencedColumnName="pack", nullable=false, insertable=true,
                                    updatable=true)})
            private Validityperiod validityperiod;
```

Entity ValidityperiodsalesreportPK

```
@Embeddable
public class ValidityperiodsalesreportPK implements Serializable {
            private static final long serialVersionUID = 1L;
            @Column(insertable=false, updatable=false, unique=true, nullable=false, length=2)
            private String monthnumber;
            @Column(insertable=false, updatable=false, unique=true, nullable=false)
            private long pack;
```

TelcoConsumerWEB Components

- Servlets
  - BuyService
  - Confirmation
  - DetailServicePackage
  - Homepage
  - LogIn
  - Logout
  - OrderPack
  - PurchasedOrders
  - RetryPurchase
  - ServiceType
  - SignUp
- Pages
  - BuyService.jsp
  - Confirmation.jsp
  - DetailServicePackage.jsp
  - Homepage.jsp
  - PurchasedOrders.jsp
  - ServiceType.jsp
  - index.html

TelcoEmployeeWEB Components

- Client components
  - CreateFixedInternet
  - CreateMobileInternet
  - CreateMobilePhone
  - CreateOptionalProduct
  - CreatePackage
  - DetailServicePackage
  - Homepage
  - Login
  - Logout
  - SalesReport
  - ViewPackage
- Views
  - DetailServicePackage.jsp
  - Homepage.jsp
  - Login.jsp
  - SalesReport.jsp
  - ViewPackage.jsp
  - index.jsp

TelcoEJB Components

Client components

Manager:

LogInManager:

```
// check username and password entered for consumer, true if user exist

public boolean validateUser(String username, String pass);


// check username and password entered for an employee, true if user exist

public boolean validateEmployee(String username, String pass);
```

OptionalProductManager:

```
// Returns the list of all optional product

public List<Optionalproduct> allOptionalProduct();


// Add a Optional Product

public void addOptionalProduct(String name, float fee);
```

PackageManager:

```
// Returns the list of all packages

public List<Pack> allPackage();


// Add a package

public void addPackage(String name, float fee12, float fee24, float fee36, String feature1, String feature2,String feature3, Hashtable<Service, Integer> service, List<Optionalproduct> optionalProduct);


// Filter packages by service type

public List<Pack> getPackages(String type);


// Returns quantity of package-service

public int quantity(long packId, long servId);
```

PurchaseManager:

```
// Returns the list of all purchases

public List<Purchase> allPurchases();
```

```
// Create a new purchase

public void createPurchase(Date endperiod,
Date startperiod, Timestamp timestamp,
String username, Pack pack,
List<Optionalproduct> products, Float price,
int rejected);


// Retry to purchase an order

public void retryPurchase(long id, boolean
status);


SalesReportManager:

// Returns the list of all Usersalesreport

public List<Usersalesreport>
allUsersalesreport();



// Returns the list of all
Validityperiodsalesreport

public List<Validityperiodsalesreport>
allValidityperiodsalesreport();



// Returns the list of all
Purchasesalesreport

public List<Purchasesalesreport>
allPurchasesalesreport();

 // Returns the list of all Packsalesreport

public List<Packsalesreport>
allPacksalesreport();
```

```
// Returns the list of all
Optionalproductsalesreport

public List<Optionalproductsalesreport>
allOptionalproductsalesreport();


ServiceManager:

// Returns the list of all services

public List<Service> allService();


// Returns the list of all mobile phone
services

public List<Service> allMobilePhoneService();


// Returns the list of all mobile internet
services

public List<Service>
allMobileInternetService();


// Returns the list of all fixed internet
services

public List<Service> allFixedInternetService();


// Add a Mobile Phone offer

public void addMobilePhone(String name, int
min, int sms, float extraMinute, float
extraSms);


// Add a Mobile Internet offer

    public void addMobileInternet(String name,
int giga, float fee);
```

```
// Add a Fixed Internet offer

public void addFixedInternet(String name,
int giga, float fee);
```

SessionManager:

```
// checks the validity of the AccessSession
cookie and returns the associated

// username,

// or null if the session does not exist

public String user(Cookie[] cookies);


// extracts the AccessSession cookie code,
or returns -1 if it doesn't exist

public long session(Cookie[] cookies);


// generates a new SessionAccess value to be
associated with User u

// createSessionID

public long getSessionID(String username);


// delete the SessionAccess value

public void deleteSessionID(Cookie[]
cookies);
```

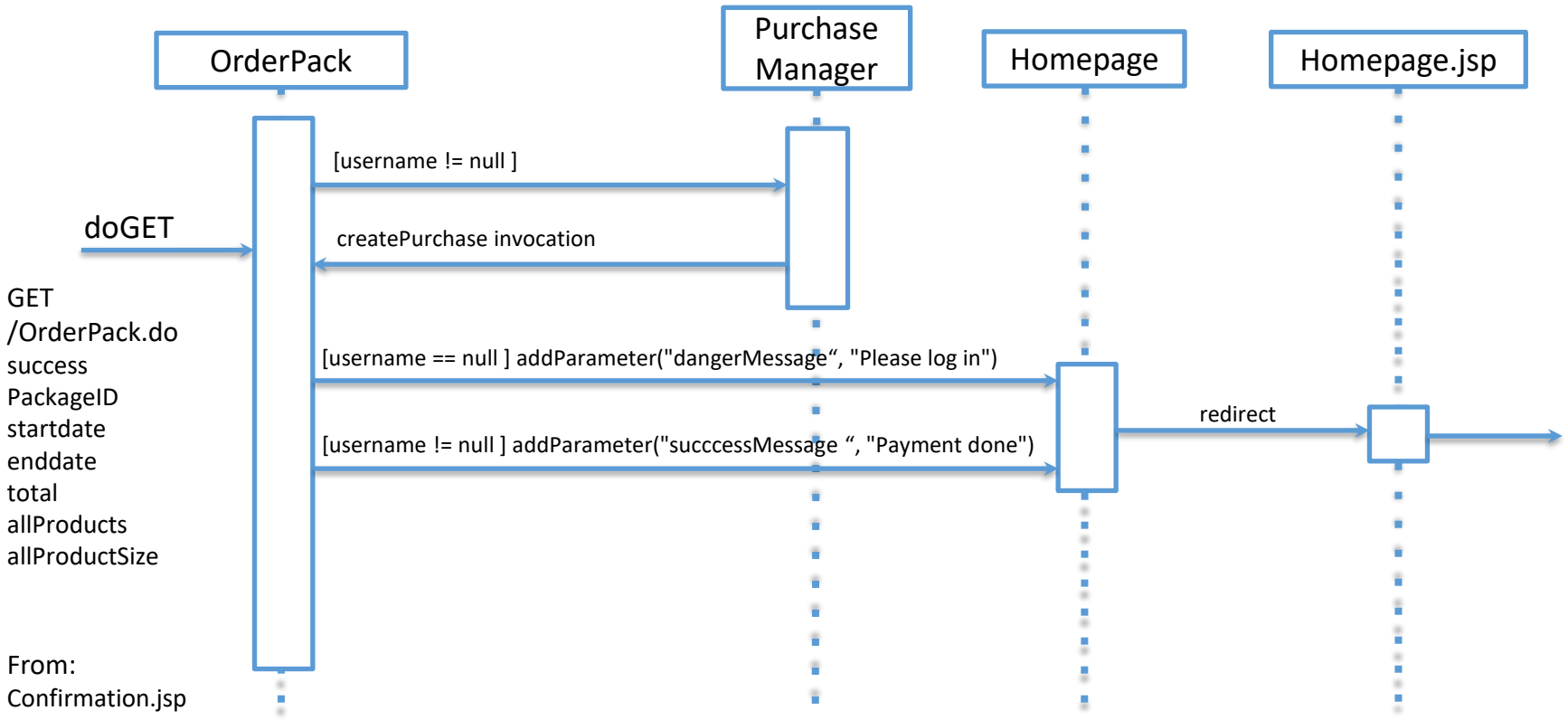
SignUpManager:

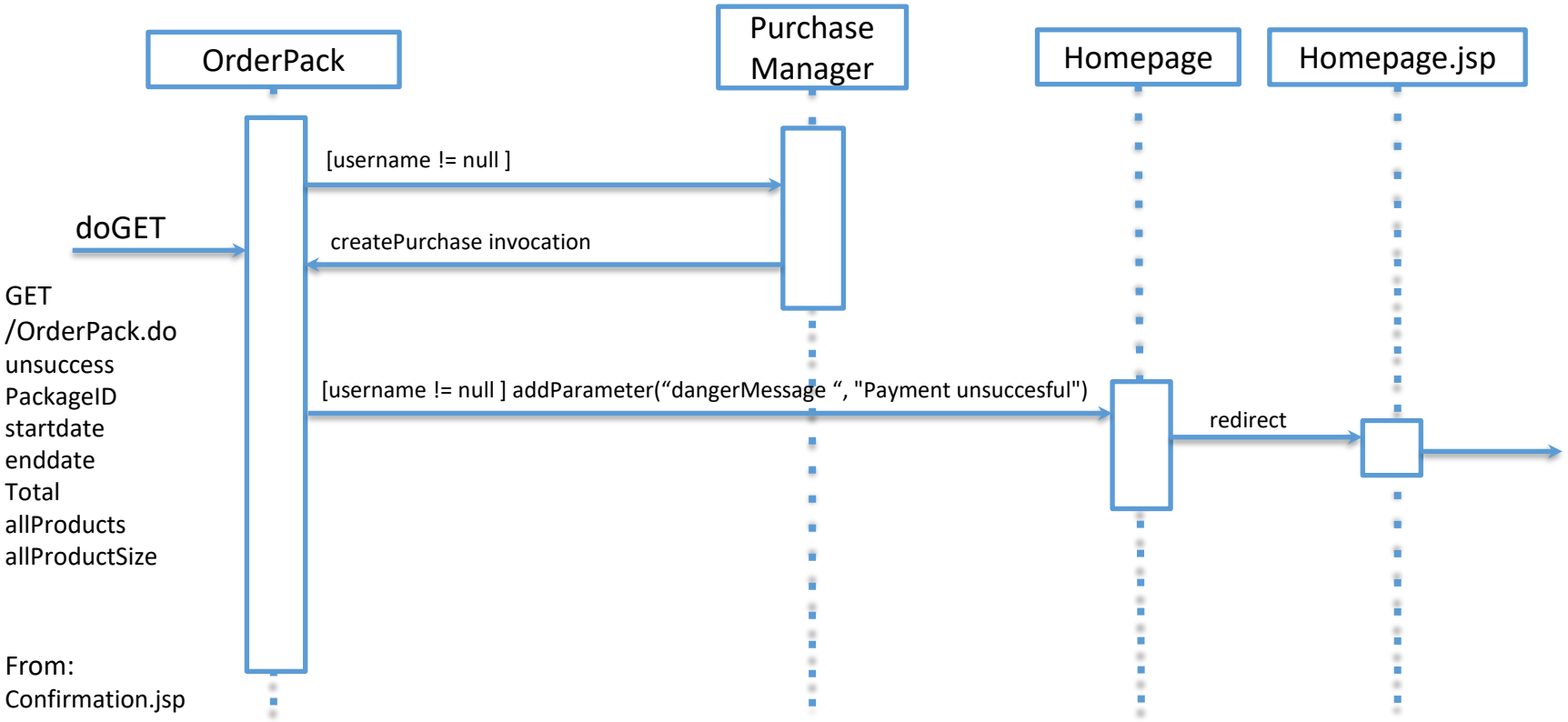
```
// add a new user

public void addUser(String username, String
name, String last_name, String role, boolean
solvent, String password,String email);
```

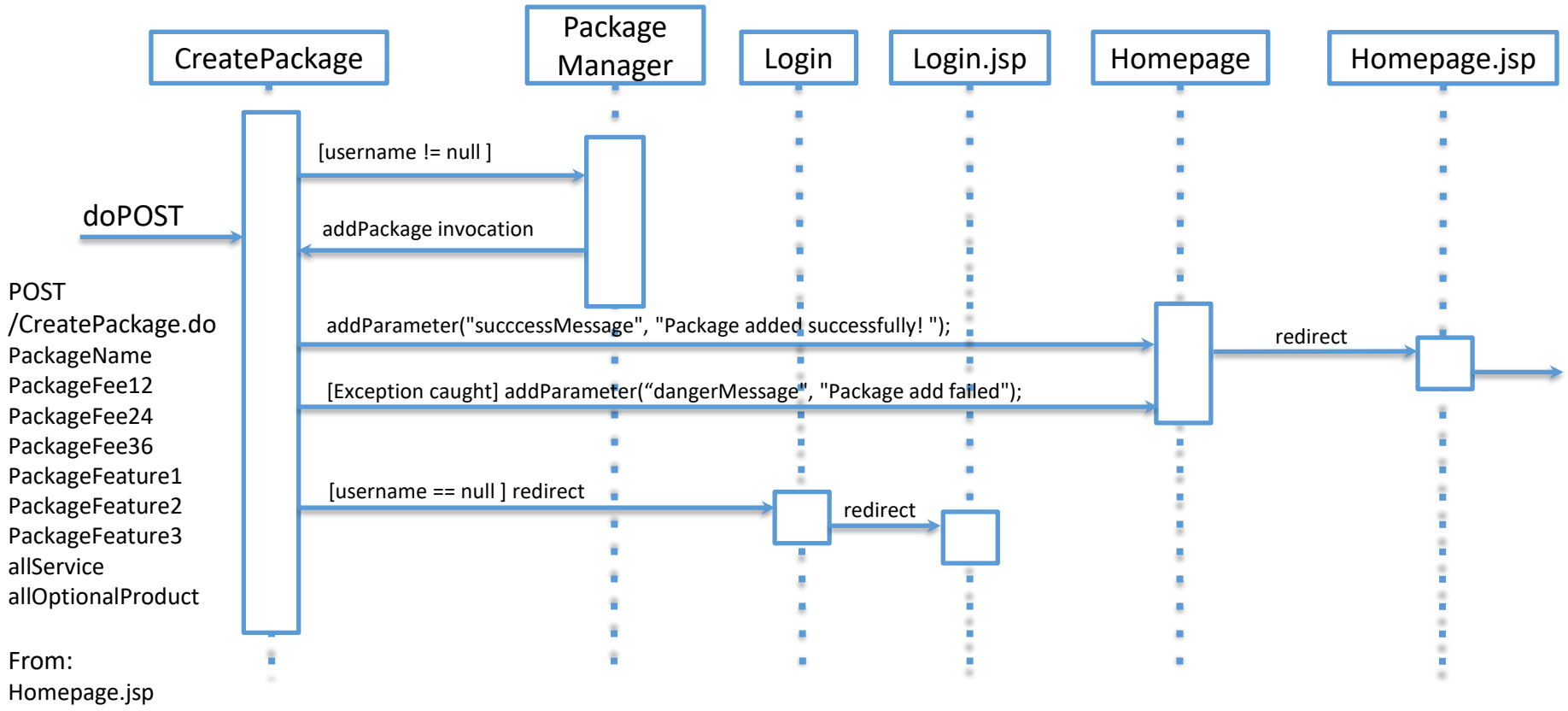
ViewDataManager:

```
// Returns the list of all packages of a type

public List<Pack> getPackages(String type);


// Compute the total cost of a purchase

public float computeCost(Optionalproduct
product, Validityperiod period);


// Returns the list of all users

public List<User> allUsers();


// Returns the list of all alerts

public List<Alert> allAlerts();
```

UML sequence diagrams




OrderPack
Purchase Manager
Homepage
Homepage.jsp
[username != null]
doGET
createPurchase invocation
GET
/OrderPack.do
unsuccess
PackageID
startdate
enddate
Total
allProducts
allProductSize
[username != null] addParameter(“dangerMessage “, "Payment unsuccesful")
redirect
From:
Confirmation.jsp


CreatePackage
Package Manager
Login
Login.jsp
Homepage
Homepage.jsp
doPOST
POST
/CreatePackage.do
PackageName
PackageFee12
PackageFee24
PackageFee36
PackageFeature1
PackageFeature2
PackageFeature3
allService
allOptionalProduct
From:
Homepage.jsp
[username != null]
addPackage invocation
addParameter("succcessMessage", "Package added successfully! ");
[Exception caught] addParameter(“dangerMessage", "Package add failed");
redirect
[username == null] redirect
redirect