



**POLITECNICO**  
MILANO 1863

## Project DB2

Telco App

**Luca Genoni - 10520445**

June 4, 2022

- ① Specification
  - Annotation
  - Additional specifications
- ② Conceptual and logical data models
  - ER diagram
  - logical model
- ③ Trigger design and code
- ④ ORM relationship design with explanations
- ⑤ Entities code
- ⑥ Interaction diagrams or functional analysis of the specifications
- ⑦ List of components
  - Back-End components
  - Front-End components
- ⑧ UML sequence diagrams



# Specification

## TELCO SERVICE APPLICATIONS

A telco company offers pre-paid online services to web users. Two client applications using the same database need to be developed.

### CONSUMER APPLICATION

The **consumer** application has a public Landing page with a form for login and a form for registration. Registration requires a **username** (which can be assumed as the unique identification parameter), a **password** and an **email**.

Login leads to the Home page of the consumer application. Registration leads back to the landing page where the user can log in.

The user can log in before browsing the application or browse it without logging in. If the user has logged in, his/her username appears in the top right corner of all the application pages.

# Packages

The Home page of the consumer application displays the service packages offered by the telco company.

A **service package** has an **ID** and a **name** (e.g., “Basic”, “Family”, “Business”, “All Inclusive”, etc). **It comprises one or more services**. **Services** are of four types: fixed phone, mobile phone, fixed internet, and mobile internet. The mobile phone service specifies the **number of minutes** and **number of SMSs** included in the package plus the **fee for extra minutes** and the **fee for extra SMSs**. The fixed phone service has no specific configuration parameters. The mobile and fixed internet services specify the **number of Gigabytes** included in the package and the **fee for extra Gigabytes**.

# Packages relations

A service package must be associated with one validity period . A validity period specifies the number of months (12, 24, or 36). Each validity period has a different monthly fee (e.g., 20€/month for 12 months, 18€/month for 24 months, and 15€/month for 36 months).

A service package may be associated with one or more optional products (e.g., an SMS news feed, an internet TV channel, etc.). The validity period of an optional product is the same as the validity period that the user has chosen for the service package. An optional products has a name and a monthly fee independent of the validity period duration. The same optional product can be offered in different service packages.

# Buy page

From the Home page, the user can access a Buy Service page for purchasing a service package and thus creating a service **subscription**. The Buy Service page contains a form for purchasing a service package. The form allows the user to select one service package from the list of available ones and choose the validity period duration and the optional products to buy together with the chosen service. The form also allows the user to select the **start date** of his/her subscription. After **choosing the service packages, the validity period and (0 or more) optional products**, the user can press a CONFIRM button.

The application displays a CONFIRMATION page that summarizes the details of the chosen service package, the validity period, the optional products and the **total price** to be pre-paid:  $(\text{monthly fee of service package} * \text{number of months}) + (\text{sum of monthly fees of options} * \text{number of months})$ .

If the user has already logged in, the CONFIRMATION page displays a BUY button. If the user has not logged in, the CONFIRMATION page displays a link to the login page and a link to the REGISTRATION page. After either logging in or registering and immediately logging in, the CONFIRMATION page is redisplayed with all the confirmed details and the BUY button.



# Order & successful payment

When the user presses the BUY button, an order is created. The **order** has an **ID** and a **date and hour of creation**. It is associated with the **user** and with the **service package**, its **validity period** and the **chosen optional products**. It also contains the **total value** (as in the CONFIRMATION page) and the **start date** of the subscription. After creating the order, the application bills the customer by calling an external service. If the external service accepts the **billing**, the order is marked as **valid** and a service activation schedule is created for the user. A **service activation schedule** is a record of the **services** and **optional products** to activate for the **user** with their **date of activation** and **date of deactivation**.

# After-effect of failed payment

If the external service rejects the billing, the order is put in the **rejected status** and the user is flagged as **insolvent**. When an insolvent user logs in, the home page also contains the list of rejected orders.

The user can select one of such orders, access the CONFIRMATION page, press the BUY button and attempt the payment again. When the same user causes three failed payments, an **alert** is created in a dedicated auditing table, with **the user ID**, **username**, **email**, and the **amount**, **date and time of the last rejection**.  
[...]

# After-effect of failed payment

If the external service rejects the billing, the order is put in the **rejected status** and the user is flagged as **insolvent**. When an insolvent user logs in, the home page also contains the list of rejected orders.

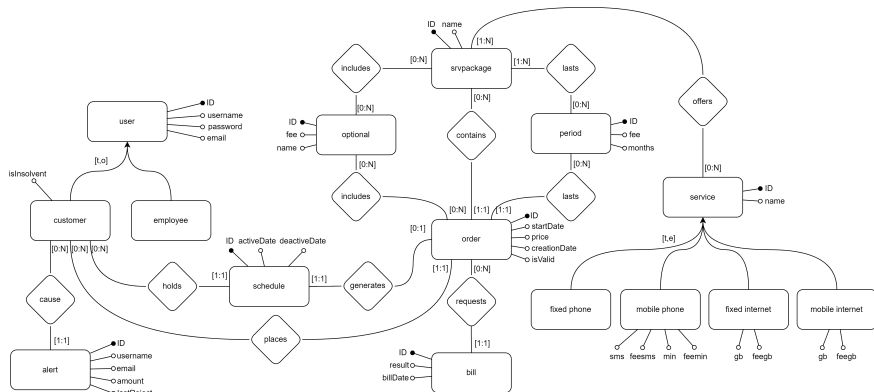
The user can select one of such orders, access the CONFIRMATION page, press the BUY button and attempt the payment again. When the same user causes three failed payments, an **alert** is created in a dedicated auditing table, with **the user ID**, **username**, **email**, and the **amount**, **date and time of the last rejection**.  
[...]

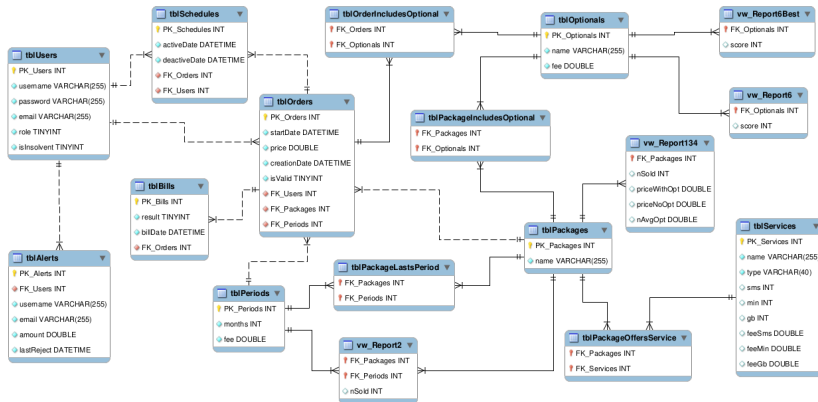
# My assumptions

- Employee are also customer
- Subscriptions are just orders without a set user.
- A bill store the response from the payment gateway.
- An alert is generated with 3 or more reject bills of non-valid orders.
- An alert amount is the sum of prices of non-valid orders.
- The alert table is write-only for future inspection.



# Conceptual and logical data models







# Trigger design and code



## TR\_tblPackages\_Insertvw\_Report134

- E: insert packages
- A: insert vw\_Report134
  - ▶ After for each row

```

1 create trigger TR_tblPackages_Insertvw_Report134
2 after insert on tblPackages for each row
3 begin
4     insert into vw_Report134 (FK.Packages) values (new.
5         PK.Packages);
6 end;

```

## TR\_tblOrders\_Updatevw\_Report134

- E: update order
- C: is valid
- A: update vw\_Report134
  - ▶ After for each row

```

1 create trigger TR_tblOrders_Updatevw_Report134
2 after update on tblOrders for each row
3 begin
4     declare var_nSold int;
5     declare var_nOpt int;
6     declare var_months int;
7     declare var_fee double;
8
9     if new.isValid = 1 then
10         select count(*) into var_nSold from tblOrders where FK.Packages = new.
11             FK.Packages and isValid = 1;
12         select count(*) into var_nOpt from tblOptionals opt,
13             tblOrderIncludesOptional oio where oio.FK.Orders = new.PK.Orders and
14             opt.PK.Optionals = oio.FK.Optionals;
15         select months, fee into var_months, var_fee from tblPeriods where
16             PK.Periods = new.FK.Periods;
17         update vw_Report134 set nSold = var_nSold
18             , priceWithOpt = priceWithOpt + new.price
19             , priceNoOpt = priceNoOpt + (var_fee * var_months)
20             , nAvgOpt = ((nAvgOpt*(var_nSold-1) + var_nOpt)/(var_nSold)
21             where FK.Packages = new.FK.Packages;
22     end if;
23 end;

```

## TR\_tblPackageLastsPeriod\_Insertvw\_Report2

- E: insert period of a package
- A: insert vw\_Report2
  - ▶ After for each row

```
1 create trigger TR_tblPackageLastsPeriod_Insertvw_Report2
2 after insert on tblPackageLastsPeriod for each row
3 begin
4     insert into vw_Report2 (FK_Packages, FK_Periods) values
5         (new.FK_Packages, new.FK_Periods);
6 end;
```

## TR\_tblOrders\_Updatevw\_Report2

- E: update order
- C: is valid
- A: update vw\_Report2
  - ▶ After for each row

```
1 create trigger TR_tblOrders_Updatevw_Report2
2 after update on tblOrders for each row
3 begin
4     if new.isValid = 1 then
5         update vw_Report2 set nSold = nSold + 1
6         where FK_Packages = new.FK_Packages and
7             FK_Periods = new.FK_Periods;
8     end if;
9 end;
```

## Report6

## TR\_tblOptions\_Insertvw\_Report6

- E: insert optional
- A: insert vw\_Report6
  - ▶ After for each row

```

1 create trigger TR_tblOptions_Insertvw_Report6
2 after insert on tblOptions for each row
3 begin
4     insert into vw_Report6 (FK.Options) values (new.
5         PK.Options);
6 end;

```

## TR\_tblOrders\_Updatevw\_Report6

- E: update order
- C: is valid
- A: update vw\_Report6, delete/insert vw\_Report6Best
  - ▶ After for each row

```

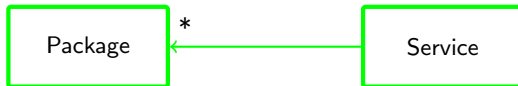
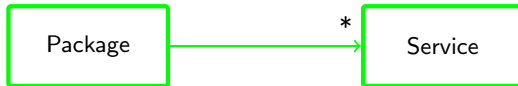
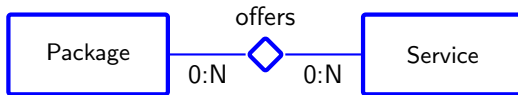
1 create trigger TR_tblOrders_Updatevw_Report6
2 after update on tblOrders for each row
3 begin
4     declare var_months int;
5     declare var_fee double;
6
7     if new.isValid = 1 then
8         select months into var_months from tblPeriods where PK.Periods = new.
9             FK.Periods;
10        delete from vw_Report6Best where score = (select Max(r.score) from
11            vw_Report6 r);
12        update vw_Report6 r
13            inner join tblOptions o on r.FK.Options = o.PK.Options
14            inner join tblOrderIncludesOptional oio on r.FK.Options = oio.
15            FK.Options
16            set r.score = r.score + (o.fee*var_months)
17            where oio.FK.Orders = new.PK.Orders;
18        if var_max > 0 then
19            insert into vw_Report6Best (FK.Options, score)
20                select *
21                from vw_Report6
22                where score = (select Max(r.score) from vw_Report6 r);
23        end if;
24    end if;
25 end;

```



# ORM relationship design with explanations

# Package-Service



Package → Service

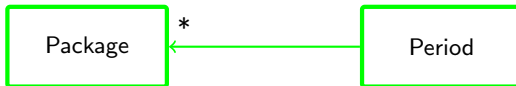
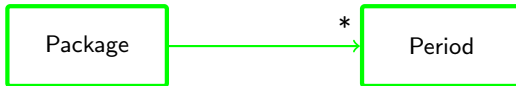
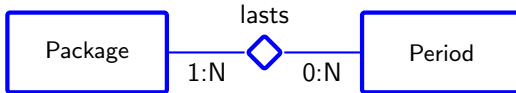
@ManyToMany

- ▶ default fetch LAZY
- ▶ cascade PERSIST, MERGE

Service → Package

/

# Package-Period



Package → Period

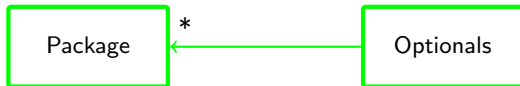
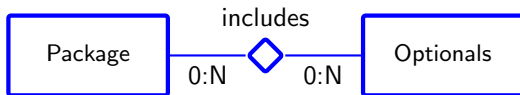
@ManyToMany

- ▶ default fetch LAZY
- ▶ cascade PERSIST, MERGE

Period → Package

/

# Package-Optionals



Package → Optionals

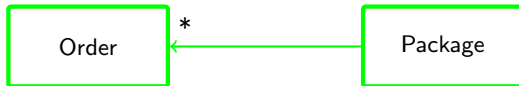
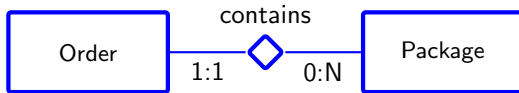
@ManyToMany

- ▶ default fetch LAZY
- ▶ cascade PERSIST, MERGE

Optionals → Package

/

## Order-Package



Order → Package

@ManyToOne

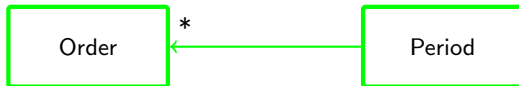
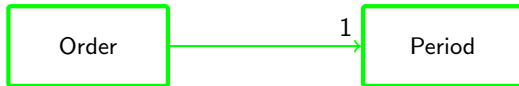
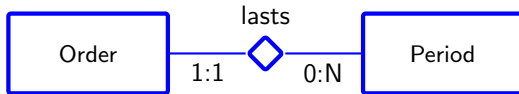
► fetch LAZY

Package → Order

/



## Order-Period



Order → Period

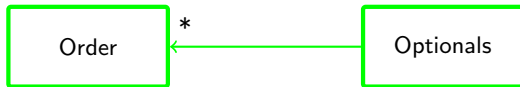
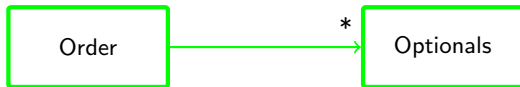
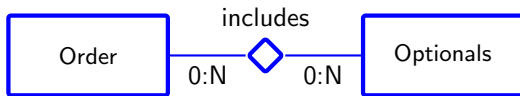
@ManyToOne

► fetch LAZY

Period → Order

/

# Order-Optionals



Order → Optionals

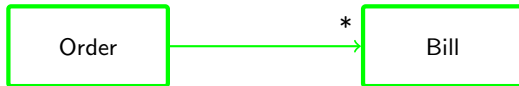
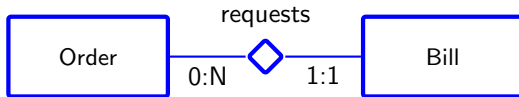
@ManyToMany

▶ default fetch LAZY

Optionals → Order

/

## Order-Bill



Order → Bill

@OneToMany

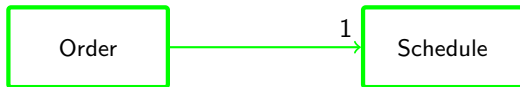
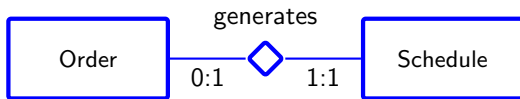
- ▶ default fetch LAZY
- ▶ cascade ALL
- ▶ orphanRemoval

Bill → Order

@ManyToOne

- ▶ fetch LAZY

## Order-Schedule



Order → Schedule

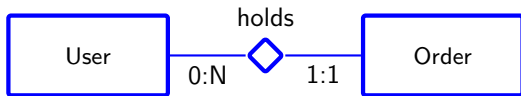
/

Schedule → Order

@OneToOne

► fetch LAZY

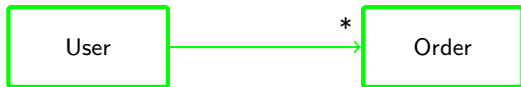
# User-Order



User → Order

@OneToMany

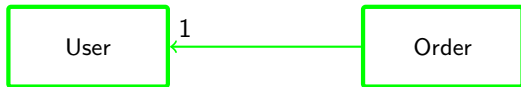
- ▶ default fetch LAZY
- ▶ cascade ALL
- ▶ orphanRemoval



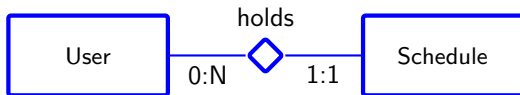
Order → User

@ManyToOne

- ▶ fetch LAZY



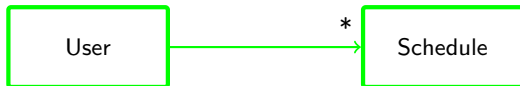
# User-Schedule



User → Schedule

@OneToMany

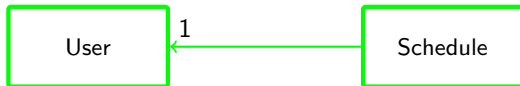
- ▶ default fetch LAZY
- ▶ cascade ALL
- ▶ orphanRemoval



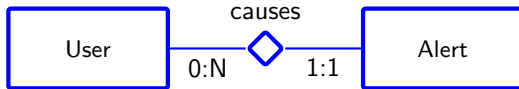
Schedule → User

@ManyToOne

- ▶ fetch LAZY



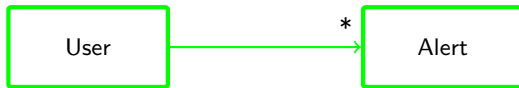
# User-Alert



User → Alert

@OneToMany

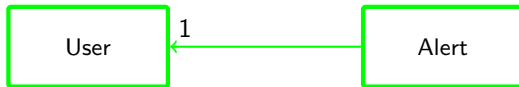
- ▶ default fetch LAZY
- ▶ cascade PERSIST, MERGE, REFRESH, DETACH



Alert → User

@ManyToOne

- ▶ fetch LAZY





## Entities code



# Entities code

## TblPackage

27/41

```
1 @Entity
2 @Table(name = "tblPackages")
3 @NamedQuery(name = "TblPackage.findAll", query = "SELECT t FROM TblPackage t")
4 @NamedQuery(name = "TblPackage.findByName", query = "SELECT t FROM TblPackage t WHERE t.name = ?1")
5 public class TblPackage implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    @Column(unique = true, nullable = false)
11    private int PK.Packages;
12
13    @Column(nullable = false, length = 255)
14    private String name;
15
16    @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
17    @JoinTable(name = "tblPackageIncludesOptional", joinColumns = {
18        @JoinColumn(name = "FK.Packages", nullable = false) }, inverseJoinColumns = {
19            @JoinColumn(name = "FK.Optionals", nullable = false) })
20    private List<TblOptional> tblOptionals;
21
22    @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
23    @JoinTable(name = "tblPackageLastsPeriod", joinColumns = {
24        @JoinColumn(name = "FK.Packages", nullable = false) }, inverseJoinColumns = {
25            @JoinColumn(name = "FK.Periods", nullable = false) })
26    private List<TblPeriod> tblPeriods;
27
28    @ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
29    @JoinTable(name = "tblPackageOffersService", joinColumns = {
30        @JoinColumn(name = "FK.Packages", nullable = false) }, inverseJoinColumns = {
31            @JoinColumn(name = "FK.Services", nullable = false) })
32    private List<TblService> tblServices;
```

```
1 @Entity
2 @Table(name = "tblServices")
3 @NamedQuery(name = "TblService.findAll", query = "SELECT t FROM TblService t")
4 public class TblService implements Serializable {
5     private static final long serialVersionUID = 1L;
6
7     @Id
8     @GeneratedValue(strategy = GenerationType.IDENTITY)
9     @Column(unique = true, nullable = false)
10    private int PK.Services;
11
12    private double feeGb;
13
14    private double feeMin;
15
16    private double feeSms;
17
18    private int gb;
19
20    private int min;
21
22    @Column(nullable = false, length = 255)
23    private String name;
24
25    private int sms;
26
27    @Column(nullable = false, length = 40)
28    private String type;
```

```
1 @Entity
2 @Table(name = "tblPeriods")
3 @NamedQuery(name = "TblPeriod.findAll", query = "SELECT t FROM TblPeriod t")
4 public class TblPeriod implements Serializable {
5     private static final long serialVersionUID = 1L;
6
7     @Id
8     @GeneratedValue(strategy = GenerationType.IDENTITY)
9     @Column(unique = true, nullable = false)
10    private int PK.Periods;
11
12    @Column(nullable = false)
13    private double fee;
14
15    @Column(nullable = false)
16    private int months;
```

```
1 @Entity
2 @Table(name = "tblOptionals")
3 @NamedQuery(name = "TblOptional.findAll", query = "SELECT t FROM TblOptional t")
4 @NamedQuery(name = "TblOptional.findByName", query = "SELECT t FROM TblOptional t WHERE t.name = ?1")
5 public class TblOptional implements Serializable {
6     private static final long serialVersionUID = 1L;
7
8     @Id
9     @GeneratedValue(strategy = GenerationType.IDENTITY)
10    @Column(unique = true, nullable = false)
11    private int PK.Optionals;
12
13    @Column(nullable = false)
14    private double fee;
15
16    @Column(nullable = false, length = 255)
17    private String name;
```

```

1  @Entity
2  @Table(name = "tblOrders")
3  @NamedQuery(name = "TblOrder.findAll", query = "SELECT t
   FROM TblOrder t")
4  @NamedQuery(name = "TblOrder.findAllRejected", query = "
   SELECT t FROM TblOrder t WHERE t.isValid=0")
5  @NamedQuery(name = "TblOrder.findAllRejectedOfUser", query = "
   SELECT t FROM TblOrder t WHERE t.isValid=0 and t.
   tblUser = ?1")
6  public class TblOrder implements Serializable {
7      private static final long serialVersionUID = 1L;
8
9      @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     @Column(unique = true, nullable = false)
12     private int PK.Orders;
13
14     @Temporal(TemporalType.TIMESTAMP)
15     @Column(nullable = false)
16     private Date creationDate;
17
18     private byte isValid;
19
20     @Column(nullable = false)
21     private double price;
22
23     @Temporal(TemporalType.TIMESTAMP)
24     @Column(nullable = false)
25     private Date startDate;

```

```

27  @OneToMany(mappedBy = "tblOrder", fetch = FetchType.LAZY,
28             cascade = CascadeType.ALL, orphanRemoval = true)
29     private List<TblBill> tblBills;
30
31  @ManyToMany
32  @JoinTable(name = "tblOrderIncludesOptional", joinColumns =
33             {
34                 @JoinColumn(name = "FK.Orders", nullable = false) },
35             inverseJoinColumns = {
36                 @JoinColumn(name = "FK.Optionals", nullable = false) })
37     private List<TblOptional> tblOptionals;
38
39  @ManyToOne(fetch = FetchType.LAZY)
40  @JoinColumn(name = "FK.Packages", nullable = false)
41     private TblPackage tblPackage;
42
43  @ManyToOne(fetch = FetchType.LAZY)
44  @JoinColumn(name = "FK.Periods", nullable = false)
45     private TblPeriod tblPeriod;
46
47  @ManyToOne(fetch = FetchType.LAZY)
48  @JoinColumn(name = "FK.Users", nullable = false)
49     private TblUser tblUser;

```

```
1 @Entity
2 @Table(name = "tblBills")
3 @NamedQuery(name = "TblBill.findAll", query = "SELECT t FROM TblBill t")
4 @NamedQuery(name = "TblBill.findAllRejectedOfRejected", query = "SELECT t FROM TblBill t WHERE t.result=0 and t.tblOrder.isValid
5 =0")
6 public class TblBill implements Serializable {
7     private static final long serialVersionUID = 1L;
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    @Column(unique = true, nullable = false)
12    private int PK_Bills;
13
14    @Temporal(TemporalType.TIMESTAMP)
15    @Column(nullable = false)
16    private Date billDate;
17
18    @Column(nullable = false)
19    private byte result;
20
21    @ManyToOne(fetch = FetchType.LAZY)
22    @JoinColumn(name = "FK_Orders", nullable = false)
23    private TblOrder tblOrder;
```

```
1 @Entity
2 @Table(name = "tblSchedules")
3 @NamedQuery(name = "TblSchedule.findAll", query = "SELECT t FROM TblSchedule t")
4 public class TblSchedule implements Serializable {
5     private static final long serialVersionUID = 1L;
6
7     @Id
8     @GeneratedValue(strategy = GenerationType.IDENTITY)
9     @Column(unique = true, nullable = false)
10    private int PK.Schedules;
11
12    @Temporal(TemporalType.TIMESTAMP)
13    @Column(nullable = false)
14    private Date activeDate;
15
16    @Temporal(TemporalType.TIMESTAMP)
17    @Column(nullable = false)
18    private Date deactiveDate;
19
20    @OneToOne(fetch = FetchType.LAZY)
21    @JoinColumn(name = "FK.Orders", nullable = false)
22    private TblOrder tblOrder;
23
24    @ManyToOne(fetch = FetchType.LAZY)
25    @JoinColumn(name = "FK.Users", nullable = false)
26    private TblUser tblUser;
```

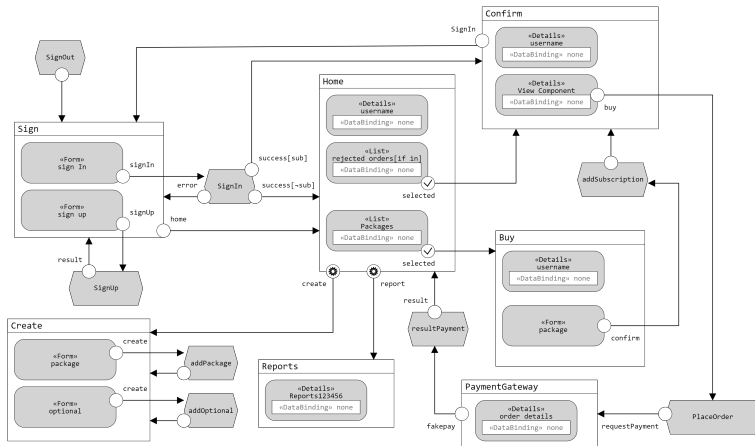
```
1 @Entity
2 @Table(name = "tblUsers")
3 @NamedQuery(name = "TblUser.findAll", query = "SELECT t FROM TblUser t")
4 @NamedQuery(name = "TblUser.checkSignIn", query = "SELECT t FROM TblUser t WHERE t.username = ?1 and t.password = ?2")
5 @NamedQuery(name = "TblUser.findAllInsolvent", query = "SELECT t FROM TblUser t WHERE t.isInsolvent = 1")
6 @NamedQuery(name = "TblUser.findByUsername", query = "SELECT t FROM TblUser t WHERE t.username = ?1")
7 public class TblUser implements Serializable {
8     private static final long serialVersionUID = 1L;
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     @Column(unique = true, nullable = false)
13     private int PK_Users;
14
15     @Column(nullable = false, length = 255)
16     private String email;
17
18     @Column(nullable = false)
19     private byte isInsolvent;
20
21     @Column(nullable = false, length = 255)
22     private String password;
23
24     @Column(nullable = false)
25     private byte role;
26
27     @Column(nullable = false, length = 255)
28     private String username;
```



```
1 @Entity
2 @Table(name = "tblAlerts")
3 @NamedQuery(name = "TblAlert.findAll", query = "SELECT t FROM TblAlert t")
4 public class TblAlert implements Serializable {
5     private static final long serialVersionUID = 1L;
6
7     @Id
8     @GeneratedValue(strategy = GenerationType.IDENTITY)
9     @Column(unique = true, nullable = false)
10    private int PK_Alerts;
11
12    @Column(nullable = false)
13    private double amount;
14
15    @Column(nullable = false, length = 255)
16    private String email;
17
18    @Temporal(TemporalType.TIMESTAMP)
19    @Column(nullable = false)
20    private Date lastReject;
21
22    @Column(nullable = false, length = 255)
23    private String username;
24
25    @ManyToOne(fetch = FetchType.LAZY)
26    @JoinColumn(name = "FK_Users", nullable = false)
27    private TblUser tblUser;
```



# Interaction diagrams or functional analysis of the specifications





## List of components

# Model objects

## ■ Model objects (Entities - Beans)

- ▶ TblAlert
- ▶ TblBill
- ▶ TblOptional
- ▶ TblOrder
- ▶ TblPackage
- ▶ TblPeriod
- ▶ TblSchedule
- ▶ TblService
- ▶ TblUser
- ▶ Vw\_Report134
- ▶ Vw\_Report2
- ▶ Vw\_Report2PK
- ▶ Vw\_Report6Best

# Data access objects

## ■ UserService - Stateless

- ▶ createUser(String username, String email, String password, int role)
- ▶ checkCredentials(String usrn, String pwd)
- ▶ checkIfEmployee(int idUser)
- ▶ findUser(int idUser)
- ▶ findAllRejectedOrdersOfUser(int idUser)

## ■ OrderService - Stateless

- ▶ createSubscription(Date startDate, int idperiod, int idsrvpkg, List<Integer> idopts)
- ▶ createOrder(TblUser u, TblOrder o)
- ▶ findRejectOrder(int ido, int idUser)
- ▶ createBill(int ido, byte result)

# Data access objects

## Data access objects (Business Components - EJBs)

### ■ ProductService - Stateless

- ▶ createSrvpackage(String name, List<Integer> idservices, List<Integer> idperiods, List<Integer> idoptionals, int idUser)
- ▶ createOptional(String name, double fee, int idUser)
- ▶ findSrvpackage(int id)
- ▶ findAllPackages()
- ▶ findAllServices()
- ▶ findAllPeriods()
- ▶ findAllOptionals()

### ■ ReportService - Stateless

- ▶ findAllVw\_Report134()
- ▶ findAllVw\_Report2()
- ▶ findAllInsolventTblUser()
- ▶ findAllRejectedTblOrder()
- ▶ findAllTblAlert()
- ▶ findAllVw\_Report6Best()

# Controllers & Templates

## Controller navigation (servlet)

- GoToSignPage
- GoToHomePage
- GoToBuyPage
- GoToConfirmPage
- GoToPaymentPage
- GoToEmpCreate
- GoToEmpReports

## Controller action (servlet)

- ActionSignIn
- ActionSignUp
- ActionSignOut
- ActionAddEmpPackage
- ActionAddEmpOptional
- ActionAddSubscription
- ActionPlaceOrder
- ActionResultPayment

## Templates (Views)

- Sign
- Home
- Buy
- Confirm
- Payment
- EmpCreate
- EmpReports





# UML sequence diagrams

## Result payment

