UNIVERSIDAD POLITECNICA DE VALENCIA

Laboratory

Sessions 5, 6 7

Deliverable 1

Ingeniería del Software

ETS Ingeniería Informática
DSIC – UPV

Curso 2019-2020

1. Scope of the deliverable

In the second deliverable the students will present the work developed over the latest laboratory sessions on the implementation over the study case ***EcoScooter:***

- Session 2 and 3. OO Design. Logic Layer. Constructors and Classes Design Code Generation and Constructors
- Session 4. Persistence layer with Entity Framework
- In particular, students will have to demonstrate the implementation of:
- The business layer for all the classes of the design model presented in the bulletin of the 2nd and 3rd sessions.
- The complete programming of the **business and persistence layers** of the following use cases from Figure 1:
    - Sign-up (actor: Anonymous)
    - User Login (actor: user)
    - Employee Login (actor: Employee)
    - Register station (actor: Management Employee)
    - Scooter Registration (actor: Maintenance Employee)
    - Rent a Scooter (actor: user)
    - Return a scooter (actor: user)
    - Incident Registration (actor: user)
    - Get Routes (actor: user)º

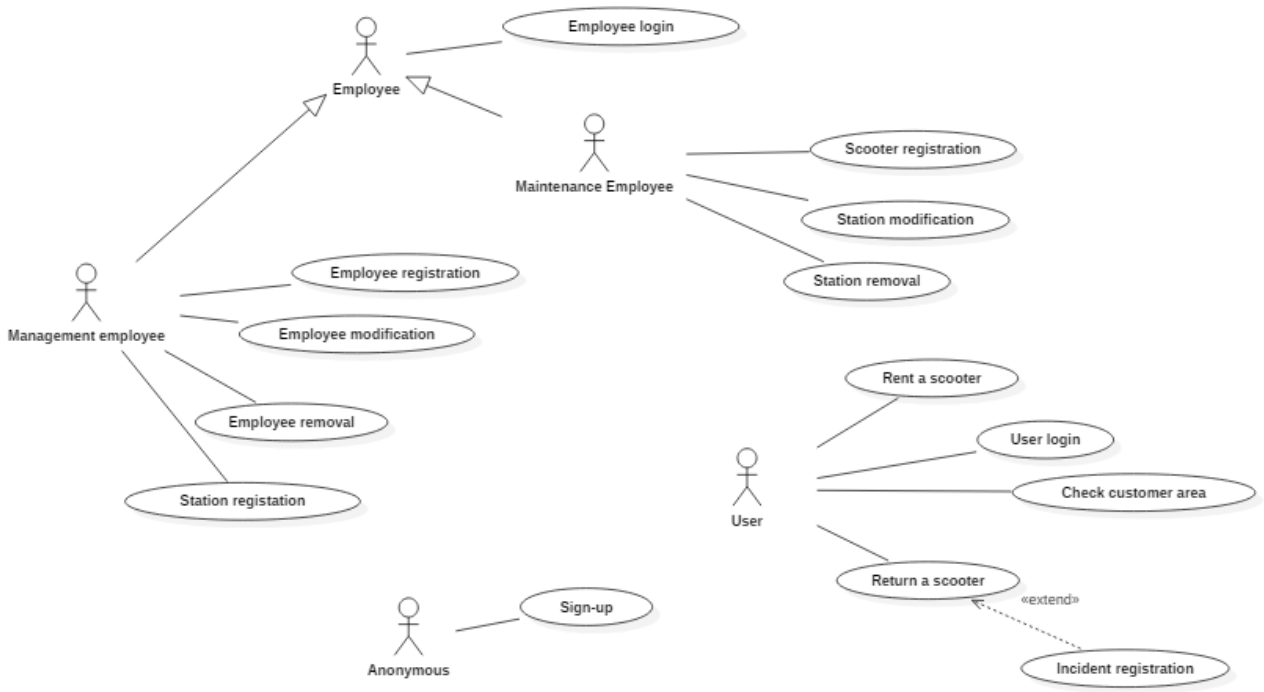


Figure 1. Partial User Case Diagram of the EcoScooter Case Study

The final code must follow the guidelines of the bulletins of sessions 2nd, 3rd and 4th sessions as well as the theory seminars)

Note: these use cases should be implemented in sessions 5, 6 and 7.

2. Description of the use cases to develop

In the next use case templates, the desired functionality of the requested use cases are described. For this deliverable, **only** the persistence and business logic is requested. The i**nterface layer is not required,** which should be delivered in the following deliverable.

ID	1
Use Case	**Sign-Up**
Actors	Anonymous
Purpose	Registers a new user into the EcoScooter System.
Summary	1. The user introduce his/her data: name, date of birth, telephon, e-mail, credit card(number, cvv, month and year of the expired date), login and password. 2. The system checks data: user over 16 years old; valid credit (8 digits and 3 digits for the cvv code; expired date in the future). If data is correct, the systems creates a new user
Precond	--
Postcond	The user is saved in the system
Synch extension	In 2, if the user provides incorrect or missing data, the system will show an error and the control goes to step 1

ID	2
Use Case	**User Login**
Actors	User
purpose	Allows user login the system.
Summary	1. The user introduces login and password 2. System verifies data and logins the user.
Precond	--
Postcond	The user is identified in the application
Synch extension	In 2, if user gives wrong data or the user doesn't exist, the system will show an error

The use case Login Employee is similar to use case 2, thus its template has not been included.

ID	3
Use Case	**Register station**
Actors	Management Employee
purpose	Add a new station in the system
Summary	1. The employee introduces the data of a new station: address (street, number and city), GPS coordinates (latitude and longitude) and the station id. 3. The system checks the input data and creates and stores a new station in the system.
Precond	The employee is logged in
Postcond	A new station is stored
Synch extension	In 2, if the employee gives wrong or missing data, or the id of the station already exists, the systems shows an error and the control goes to step 1.

ID	4
Use Case	**Scooter Registration**
Actors	Maintenance Employee
Purpose	Adds a new station to the system
Summary	1. The employee introduces the data of the scoter (register data and state). The id is generated by Entity Framework. 2. The system checks the input data. If the state is available, the system asks the station in which it will be anchor. 3. The employee introduces the station 4. System creates and stores the scooter
Precond	The employee is logged in
Postcond	A new scooter is store in the system
Synch extension	In 2, if the data is wrong, an error message is displayed and the control goes to the step 1 In 4, if the station doesn't exist, an error message is displated and the control goes to the step 3.

ID	5
Use Case	**Rents a scooter**
Actors	User
Purpose	Rents a scooter from one station
Summary	1. The user selects the station in which will made the rental 2. The systems assigns a scooter from the station (battery level is not controlled) 3. The user gets the scooter 4. The system creates and stores the new rental
Precond	The user is logged in
Postcond	The new rental is stored in the system
Synch extension	In 2, if there are not available scooters in the station, an error message is displayed

Notice that for simplify the study case, the track points, battery level and speed will not be taken into account.

ID	6
Use Case	**Return a scooter**
Actors	User
Purpose	Returns a previously rented scooter.
Summary	1. The user selects return scooter and introduces the station Id. 2. The system gets the last user rental and the serial number of the scooter 3. The system asks if any incident has happen during the rental. 4. If the was any incident, the user clicks on YES and the control goes to the use case **Incident Registration**. 5. The system updates the data of the rental: date and time of the devolution, the destination station and price. Also, it changes the state of the scooter to available.
Precond	The user is logged in
Postcond	The system stores the new data
Synch extension	In 1, if the id of the station doesn't exist, an error message is displayed In 2, if the last rental already was finish (the scooter was returned), an error message is displayed

ID	7
Use Case	**Incident Registration**
Actors	User
Purpose	Register a new incident associated to a rental
Summary	1. The user enters a description of the incident type and the date and time in which it occurred. 2. The system created the incident and associates it to the rental,
Precond	The user is logged in
Postcond	The system stores the new data
Synch extension	

ID	8
Use Case	**Get Routes**
Actors	User
Purpose	Gets a list with the routes made by the user in a period
Summary	1. The user enters the started and ended dates of the period. 2. The system retrieves the user's routes in the given period. Each route contains the following fields splitted by commas: started and ended dates and times, price, origin station id and destination station id.
Precond	The user is logged in
Postcond	
Synch extension	In 1, if the interval of dates is wrong, an error message is shown In 2, if there are not routes, an error message is shown.

3. Details about the logical layer

In previous laboratory sessions, a part of the system functionality has been build, concretely the class design and the persistence.

The controller (or service provider) of the business logic must be implemented in this deliverable. This controller must offer the upper layer (UI) access to all the functionality it needs.

All the services served to the UI should defined in an interface named `IEcoScooterService`. The methods of that interface will be implemented in a class named `EcoScooterService`. The interface and its implementation should be placed in the folder `BusinessLogic/Services`, and be part of the namespace `EcoScooter.Services`. All the errors in the implemented methods should be manage thought exceptions.

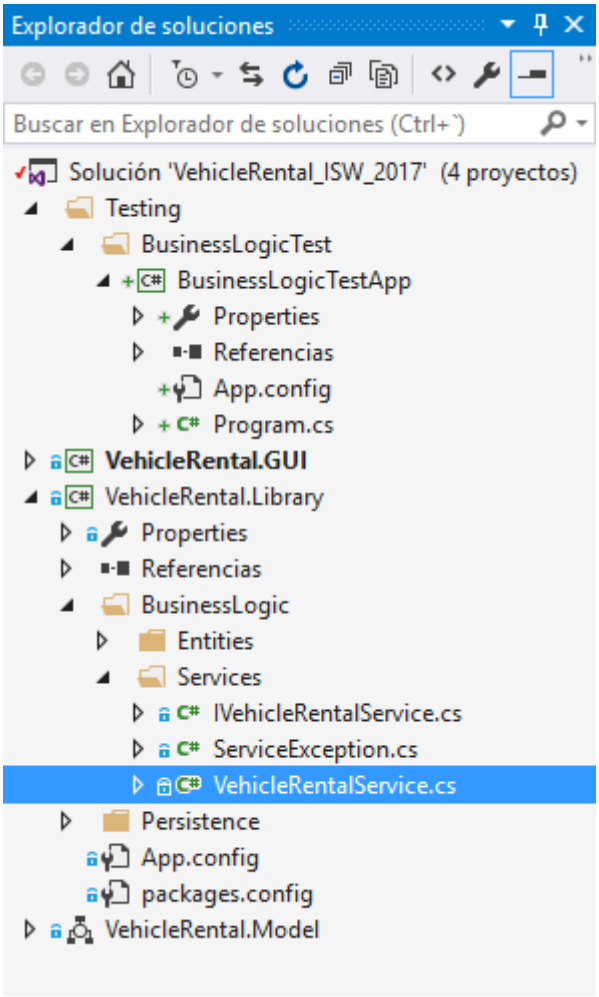
To report errors generated at the logical layer, a subclass from `Exception` named `ServiceException` should be created in the folder `BusinessLogic/Services`. Thus, the objects of this class should contain a text message with the generated error in their property `Message`.

For example, in the study case *VehicleRental_ISW* you can found in PoliformaT, an error occurs when a user tries to add a person with a dni which belongs to an already registered person. Next, the method code which add a person into the system is shown. You can see how it checks if the dni is already registered in the system previously to add the person. If it previously exists, an exception is raised. .

```
public void addPerson(Person person)
{
   if (dal.GetById<Person>(person.Dni) == null)
   {
      dal.Insert<Person>(person);
      dal.Commit();
   }
```

```
        else throw new ServiceException("Person already exists.");
    }
```

In the following figure, the solution explorer for the *VehicleRental_ISW* example is shown. Highlights the `IVehicleRentalService` interface, its implementation in a class named `VehicleRentalService`, the exception class `ServiceException` and the test program `BusinessLogicTestApp`.



In order to facilitate the method identification to implement in *BikeClubService*, you will be able to download an initial service interface description and implementation. This interface should be **extended** to the concrete **necessities** of each particular **design.**

eadline

The deadline is the 8th laboratory session **(27th November)**. **All team members** should go to the deliver session, since they may be required to answer questions about the project.

4. Delivery form

A task should be created in Poliformat, in which each group should add a zip folder with the source code of the project. Also, a new commit with the message “Deliverable 1” will be pushed in the remote repository. The project should be **attached** to the task **before** the evaluation session.

5. Evaluation

In the laboratory session, the teacher will evaluate each group, checking the correctness of the required functionality, **and asking to ALL team members**. The attached evaluation rubric will be used to assess the functionality implemented.