REQUIREMENT ANALYSIS

The following Case Study is frames into the managerial fields about a pharmacy and the products available in this. The requirement is expressed in natural language and later will be analysed and elaborated obtaining a more detailed description of the system's needs.

This section is articulated in this way: the first part will concern the analysis of the environment of use and the users; the second part will concern the functional requirements and the last part the non-functional requirements.

ANALYSIS OF THE ENVIRONMENT OF USE

The environment of use depends strictly from the application's domain. In our case we will have the management of a pharmacy, which users are represented by the employees of the pharmacy without any distinction between the figure involved in the work.

The specifics of the interactions between the user and the system will be analysed later during the database's design phase.

FUNCTIONAL REQUIREMENTS

Specified the users' categories, the functional requirements trying to identify the user's requirements giving a formal description. We can identify our functional requirements as:

ID	Name	Description
RF_1	User recording	Express the capacity of recording an employee of the pharmacy.
RF_2	User login	Express the capacity, to identify a pharmacy employee.
RF_3	Management of a product	Express the capacity to manage the products sales by the pharmacy.
RF_4	Management of a sale	Express the capacity, to manage the sales of the pharmacy.
RF_5	Management of the products	Express the capacity, to manage the products of the pharmacy.
RF_6	Management of the prescriptions	Express the capacity, to manage the prescriptions' information of the pharmacy.

RF_7	Statistical management of the sales	Express the capacity to elaborate the data of the pharmacy.

NON-FUNCTIONAL REQUIREMENTS

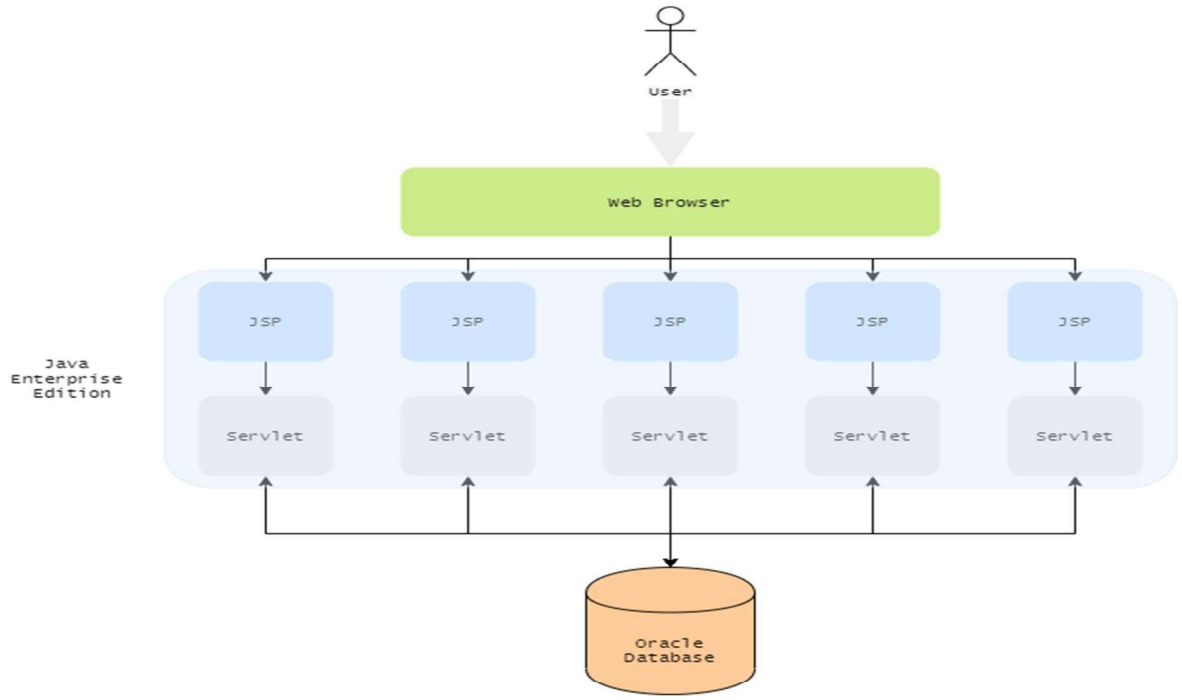
Non-functional requirements are those requirements not directly expressible. In this category we can identify the graphical layout and the usability of the system, in addition, the security requirements needed by a management application and the correctness the code. In an application like this, we will concern on the reliability and reuse capacity.

TECHONOLOGIES USED

About the technologies used we can identify at first the Oracle Database, studied during the course and which occupy the main parts of this projects. About the application and the back-end side, the chosen language is Java Enterprise Edition, deployed to the Apache Tomcat Server. The front-end side will be developed using Bootstrap 4 framework, and other technologies such as: JavaScript and AJAX.

ARCHITECTURE

According to the functional requirements identified and the technology chosen, we can identify the architecture of our application, from an abstract point of view as follow:



DATABASE DESIGN PHASE

The Database design phase is the core of the entire project, which base the entire working on it. As said before, the phase is divided in three steps according to the methodology expressed by the textbook.

The first step is the Conceptual Design, which starting from the text requirements, analysing and refining, produce a conceptual schema that sum-up and identify the main concepts. The conceptual schema adopted is the Entity-Relationship schema.

The second step is the Logical Design, which remodel the Entity-Relational schema in order to define a Logical Schema, depending on the DBMS adopted, in this case Oracle.

The last step is the Physical Design, which using a "tuning" operation defines, the physical structures that optimize the functionalities that the Database must exposes.

The following is the text representing the Database's requirements:

> We want to represent a database for the management of the products available in a pharmacy taking into account the following information: Each product is characterized by a name and description. The products in the pharmacy can be drugs or perfumery products. Perfumery products are of different types: cosmetics, hygiene products as well as baby care products.
>
> Each drug is uniquely identified by its name and by the information relating to the pharmaceutical company that produces it (the pharmaceutical company is uniquely identified by name and address). Medicines can be sold either with a prescription or without a prescription.
>
> Drugs can be covered by patents for a number of years; in this case we speak of patented or branded drugs. In addition, the pharmacy sells non-patented drugs. A non-patented drug is an unbranded drug equivalent to a patented drug. Each patented drug can have one or more equivalent non-patented drugs.
>
> The pharmacy wants to store the prescriptions received. A prescription contains a list of prescription medicines, is made by a doctor and is assigned to a patient. We want to keep track of the sales of pharmacy products, with an indication of the quantity and day of the sale and any associated prescriptions.
>
> In addition, the pharmacy is interested in carrying out statistical sales surveys and intends to memorize the semi-annual and annual reports of the sales made.

CONCEPTUAL DESIGN

The first step of the Conceptual Design consists in the requirement's analysis, consists in applying a series of rules to obtain a more precise and non-ambiguous specification.

We can try to fix some rules to obtain a more detailed specification:

- **Chose the correct level of abstraction**: it's better avoiding terms too generic or too specifics that don't make clear the concept. About the doctors (line 18), we don't have information about them, thus is better specify the information to represent, in this case we want to represent: Name, Surname and registration number of the doctor.
- **Standardize sentences' structure:** In the specific of requirements It's preferable to use the same syntactical style. For example, we can adopt a style like: "*For <concept> we represent <attributes>*".
- **Avoid complex phrases:** The definitions must be clear and simple. However, no changes are needed.
- **Identify synonyms and homonyms and standardize terms.** Synonyms are words that have the same meaning. In the text both synonyms "drugs" and "medicines" are used, it's preferable have only one term to use, in this case the term "medicines".
- **Make cross references explicit:** The lack of a reference's context makes some concepts ambiguous, in these cases it's better to explicit the context between terms. No changes are needed, however.
- **Construct a glossary of terms.** It's very useful for the comprehension and the specification of terms used, define a glossary that, for each term, containing: a brief description, possible homonyms and synonyms and other terms contained in the text.

Term	Description	Synonyms	Links
Products	Products sold by the pharmacy, they could be medicines or perfumery products.	Drugs, Medicines	Pharmaceutical Company, Prescription, Patient, Sales
Pharmaceutical Company	Company that sold a medicine to the pharmacy		Products
Prescription	Prescription given by a doctor to a patient that must buy a medicine in the pharmacy		Doctor, Products, Patient
Doctor	Doctor who make a prescription to a patient about one or a series of medicines		Prescription, Patient, Products

Sales	Sale of the pharmacy about one or a series of products		Products

With the requirements specification it's useful to collect the specification about the operations, with their relative frequencies:

ID	Description	Frequency
1	Insert a new cosmetic/drug	10 times per day
2	Insertion of a new sale	2000 times per day
3	Finding of all non-branded drugs equivalent to a given branded drug, sorted by name	2000 time per year
4	Printing of all drugs, sorted by the number of prescriptions in which they are present.	3 times per date
5	For a given doctor, generate the list of the prescriptions.	1 time per day
6	Find the Sales sold in a specific time range	5 times per day
7	Authentication of the user to the platform	10 times per day
8	Insert of a new Prescription	10 times per day
9	Print all the statistics related to the sales	1 time per month

Now, we can reorganize the phrases, grouping them by the referred concept:

General phrases about the pharmacy
We want to represent a database for the management of the products available in a pharmacy taking into account the following information:

Phrases related to the products
For each product we represent the name and the description.

Phrases related to specific type of products

The products in the pharmacy can be medicine or perfumery products. Perfumery products are of different types: cosmetics, hygiene products as well as baby care products

Phrases related to medicines

For each medicine we represent name and the information related to the pharmaceutical company that produces it (the pharmaceutical company is uniquely identified by name and address), each medicine is uniquely identified by these attributes. Medicines can be sold either with a prescription or without a prescription.

Phrases related to specific type of medicines

Medicines can be covered by patents for a number of years; in this case we speak of patented or branded drugs. In addition, the pharmacy sells non-patented medicines. A non-patented medicine is an unbranded medicine equivalent to a patented medicine. Each patented medicine can have one or more equivalent non-patented medicines.

Phrases related to prescriptions

The pharmacy wants to store the prescriptions received. For the prescription we represent the list of prescript medicines, and the doctor that assigned to a patient, for the doctor we represent name, surname and registration number.

Phrases related to sales

We want to keep track of the sales of pharmacy products, with an indication of the quantity and day of the sale and any associated prescriptions.

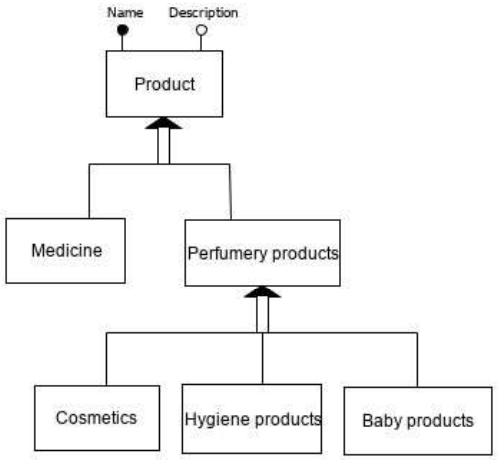
General phrases about statistics of sales

In addition, the pharmacy is interested in carrying out statistical sales surveys and intends to memorize the semi-annual and annual reports of the sales made.

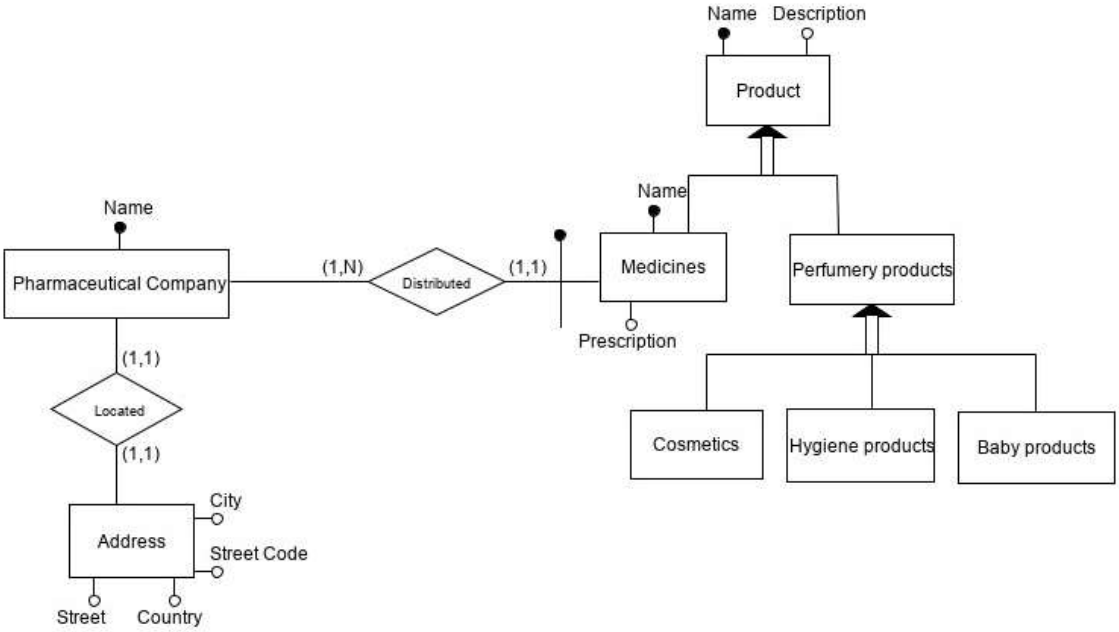
The next step of the Conceptual Design is to build the Entity-Relationship schema, however, before do this it's better decides a strategy to operate.

The chosen strategy is the *inside-out* strategy, according to this, some fundamentals concepts are individuated and then we proceed spreading outward radially.

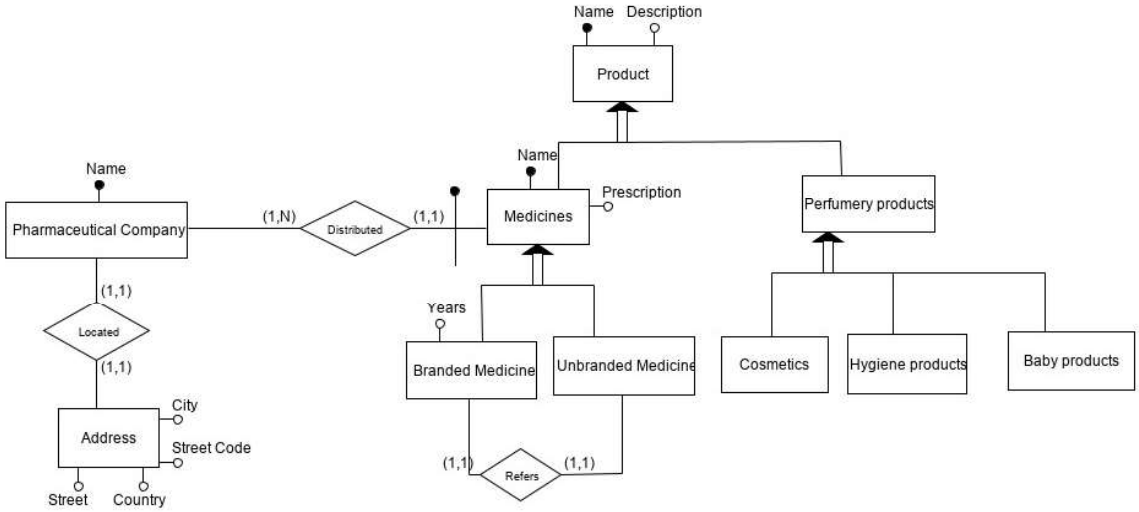
The first iteration of our strategy concerns the products sold by the pharmacy:



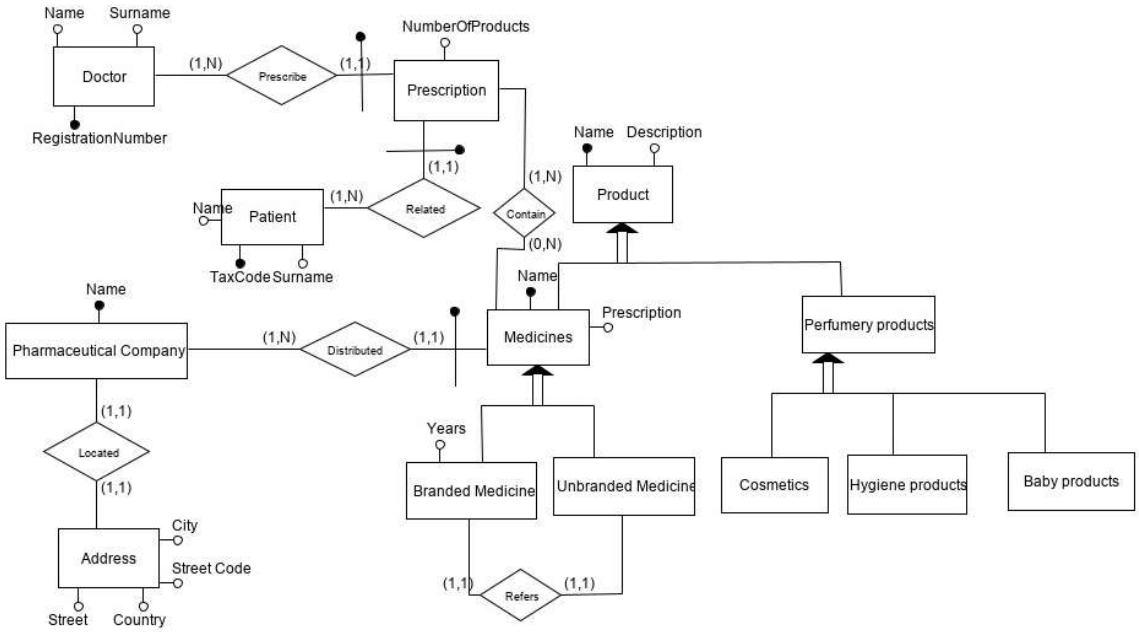
The second iteration of our strategy concerns the medicines:



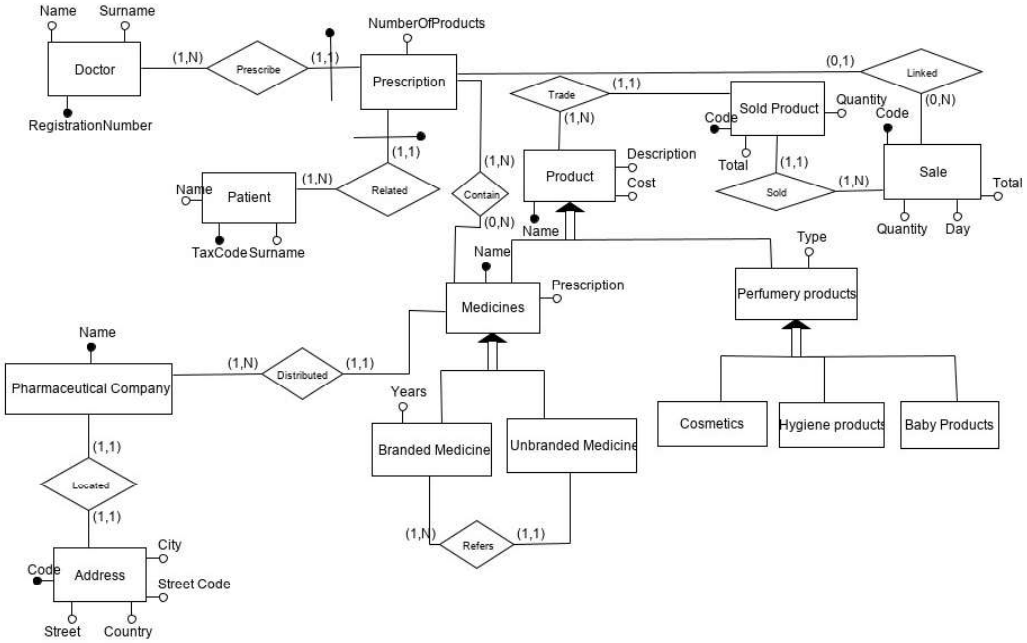
The third iteration of our strategy concerns the specific type of medicines sold:



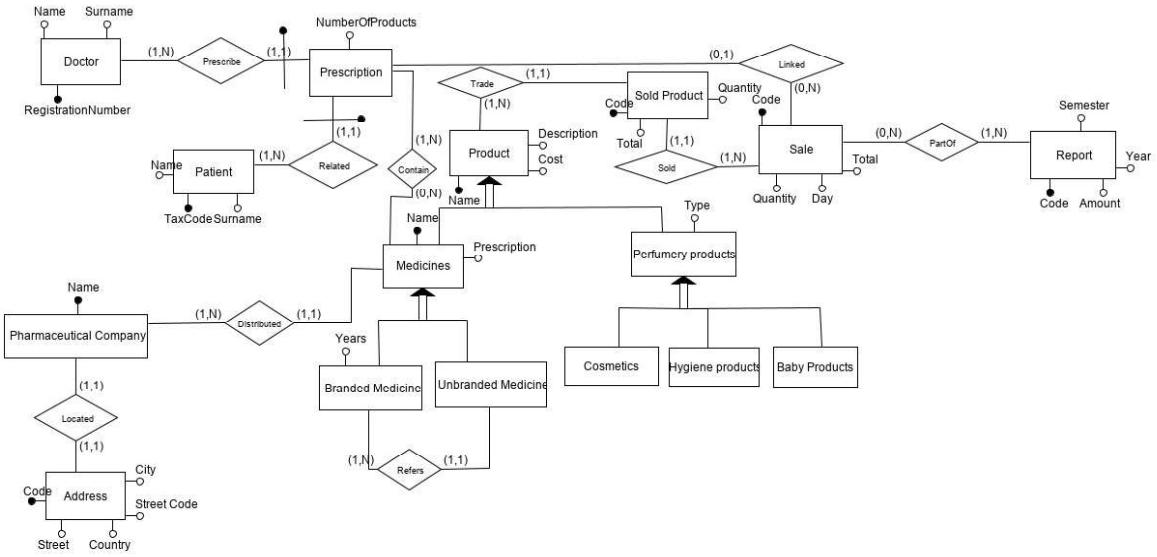
The fourth iteration of our strategy concerns the prescription made by a doctor for a patient about a medicine:



The fifth iteration of our strategy concerns the sales related to one or many products and the prescription associated to the products sold.



The last iteration of our strategy concerns the statistics associated to the sales, about this concept is necessary to add some attributes to the defined entities. The final schema derived from the iterations is the following:



To represent the concepts not directly representable by the Entity-Relationship Schema, one of the most used tools by the analysts of information systems is the *business rules*. A business rule can be:

1. A *description* of an object relevant for the application, for example the description of an entity or a relationship.
2. Some *integrity constraints*.

3. A *derivation*, thus a concept derivable from other concepts.

It's also a good usage complete the schema with a *data dictionary*, where all the concepts expressed in the schema are explained.

Entity	Description	Attributes	Identifier
Doctor	Doctor that prescribes one or some medicines to one or more patients	Name, Surname, Registration Number	Registration Number
Prescription	Prescription made by a doctor to a patient regarding some medicines	Number of products	None
Patient	Patient that requires a medicine to a doctor	Name, Surname, Tax Code	Tax Code
Address	Address of a pharmaceutical company	City, Street Code, Street, Country	None
Pharmaceutical Company	Company that sold a medicine to the pharmacy	Name	Name
Medicines	Medicine sold by the pharmacy and distributed by a pharmaceutical company	Name, Prescription	Name
Branded Medicine	Medicine branded by a pharmaceutical company	Years	None
Unbranded medicine	Medicine without brand sold by the pharmacy	None	None
Product	Product sold by the pharmacy	Name, Description, Cost	Name
Perfumery products	Perfumery products sold by the pharmacy	None	None
Cosmetics	Cosmetics sold by the pharmacy	None	None
Hygiene products	Hygiene products sold by the pharmacy	None	None
Baby products	Baby products sold by the pharmacy	None	None

Sale	Sale of some products	Quantity, Day	None
Report	Semesterly or annual report of the sales	Semester, Year, Amount	None

Relationship	Description	Entity involved	Attributes
Prescribe	Associate a specific doctor to one or more prescription	Doctor, Prescription	None
Related	Associate a Patient to one or more prescription	Patient, Prescription	None
Contain	Associate a prescription to some medicines	Prescription, Medicines	None
Located	Associate an address to a pharmaceutical company	Pharmaceutical company, address	None
Distributed	Associate a Pharmaceutical company to one medicine	Product, Pharmaceutical company	None
Refers	Associate a branded medicine to an unbranded medicine	Branded medicine, unbranded medicine	None
Sold	Associate a product to a sale	Product, Sale	None
Linked	Associate a Prescription to a Sale	Prescription, Sale	None
PartOf	Associate a sale to a report	Sale, Report	None

The integrity constraints are expressed under the form of business rules:

Constraints	
ID	Description
C_1	The Pharmaceutical Company can have only one address.
C_2	The number of years of a patent related to a medicine cannot be more than 5 for legal constraints.
C_3	The Report's year must be between 2010 and 2020.
C_4	The perfumery products must be only: "Cosmetics", "Hygiene Products", "Baby Products".
C_5	Only Medicines with prescription can be prescribed.
C_6	A Sale's date cannot be greater than current date.
Derivation rule	
DR_1	The amount of a Report is obtained summing the singles sales amount of a product.
DR_2	The amount of a Sold Product can be calculated multiplying the sold quantity for the single price

LOGICAL DESIGN

The phase of the design process is concerned on the definition of a Logical Schema according to the DBMS chose. The process is characterized on two steps:

1. **Restructuring of the Entity-Relationship schema.**
2. **Translation to a Logical Schema.**

Due to perform the first phase, we must define two tables called: table of volumes and table of operations. In the first table we define the entity and the relationship involved in the schema, defining their data load in the second table we define the operations and their frequencies.

Volumes' table		
Concept	Type	Volume
Doctor	Entity	100
Prescription	Entity	1000
Patient	Entity	500
Medicine	Entity	3000
Pharmaceutical Company	Entity	10
Address	Entity	10
Branded Medicine	Entity	2500
Unbranded Medicine	Entity	500
Perfumery	Entity	3000
Cosmetics	Entity	1000
Hygiene products	Entity	1000
Baby products	Entity	1000
Product	Entity	3000
Sale	Entity	10000
Report	Entity	177
Sold Product	Entity	20000
Prescribe	Relationship	1000
Related	Relationship	1000
Contain	Relationship	2000
Distributed	Relationship	3000
Located	Relationship	10
Refers	Relationship	500
Sold	Relationship	20000
Trade	Relationship	20000
Linked	Relationship	5000

PartOf	Relationship	56

Operations' table		
Operation	Type	Frequency
1	I	10 times per day
2	I	2000 times per day
3	I	2000 times per year
4	I	3 times per month
5	I	1 time per month
6	I	5 times per day
7	I	5 times per day
8	I	10 times per day
9	I	1 time per month

The first phase of the Logical Design is the **redundancies analysis**, consisting of analysis of redundancies and relative operations, deciding if keep the redundancies or deleting them in order to improve the performances in terms of space and workload.

The first redundancy contained in the schema, concerns the amount of the sold products of a sale, in fact we can calculate this quantity summing the single quantity of each product multiplied for the cost of it. However, we can calculate the cost of the redundancy in term of space and access. About the space we can assume that the space occupation of the attribute total is 4 bytes, for a total of:

*Space occupation = 4byte*20000 = 80.000 bytes*

To calculate the access required to obtain the attribute, is necessary to analyse the schema:

Access table in the presence of redundancy			
Concept	Construct	Access	Type
Sold Product	Entity	1	R
Sold Product	Entity	1	W

Access table without the redundancy			
Concept	Construct	Access	Type
Sold Product	Entity	1	R
Trade	Relationship	1	R
Product	Entity	7	R
Sold Product	Entity	1	W

Without the redundancy we must perform 11 accesses, compared to the 3 accesses needed for calculate the total without the redundancy. We can conclude that we kept the redundancy, in order to decrease to workload of the Database.

The second redundancy contained in the schema, concerns the amount of a Report, which can be calculated from the entity Sale whose date is contained in the time rage of the Report.

No operations are involved in this redundancy; let's calculate the cost of the redundancy. About the space we can assume that the space occupation of the attribute is:

*Space occupation = 8byte * 177 = 1416byte*

To calculate the access required to obtain the attribute, is necessary to analyse the schema:

Access table in the presence of redundancy			
Concept	Construct	Access	Type
Report	Entity	1	W

Access table without the redundancy			
Concept	Construct	Access	Type
Report	Entity	1	W
Report	Entity	1	R
PartOf	Relationship	1	R
Sales	Entity	56	R

We can conclude that: in presence of redundancy only 1*2 = 2(considering the writing access double respect to the reading) accesses are needed, without the redundancy 56 + 1+ 1 + 2 = 60 accesses are needed, to couple a space waste for 1416byte. Therefore, we kept the redundancy.

The third redundancy contained in the schema, concerns the quantity of the products sold in a Sale, which can be calculated counting the Sold Product involved in the Sales. Let's analyse the cost of the redundancy:

*Space occupation = 4byte * 10.000 = 40.000 bytes*

To calculate the access required to obtain the attribute, is necessary to analyse the schema:

Access table in the presence of redundancy			
Concept	Construct	Access	Type
Sale	Entity	1	R
Sale	Entity	1	W

Access table without the redundancy			
Concept	Construct	Access	Type
Sale	Entity	1	R
Sold	Relationship	2	R
Sold product	Entity	1	R
Sale	Entity	1	W

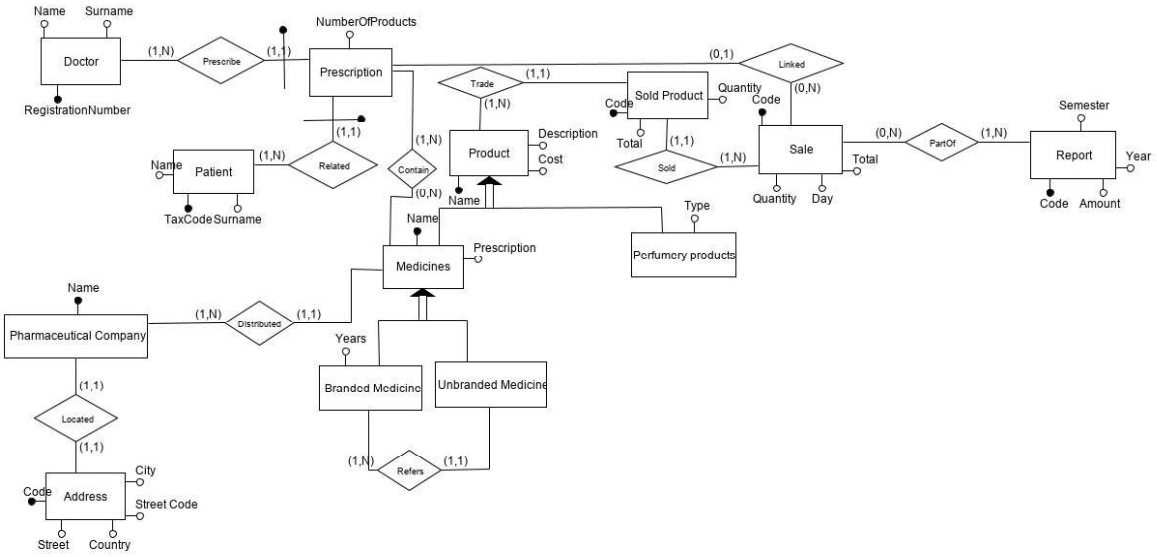
We can conclude that: in presence of redundancy only 1+2 = 3(considering the writing access double respect to the reading) accesses are needed, without the redundancy 1 + 2 +1 +2 = 6 accesses are needed, to couple a space waste for 40.000 bytes. Therefore, we kept the redundancy.

The second step is the **deleting of generalization**. It's important to note that the DBMS chosen, Oracle, allow the definition of generalization, however, even if Oracle supports the generalization, analysing the Conceptual schema we can merge the generalization about the Perfumery Products introducing an attribute that able us to distinguish the specific attribute.

The third step is the **partitioning/merging of the entity and association**. According to the schema and the operation defined, there's no need of partitioning or merging some elements in the schema.

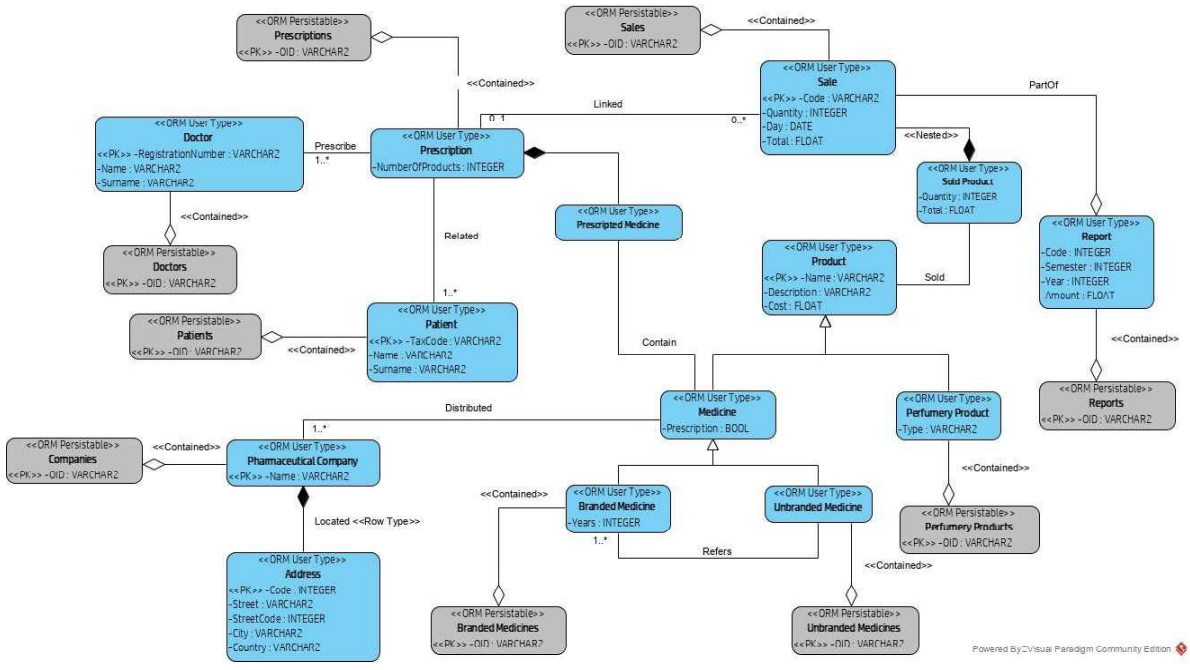
The last step is the **choice of main identificatory**, analysing the defined schema and the relative documentation produced, we must add some identifiers to the entities: Sale, Report and Address.

Applying the following updates to our Conceptual schema, we obtained the following updated schema:



After the updating of the Conceptual schema, we can conclude the first phase of the Logical Design and proceeding defining the Logical Schema.

Before starting, it's important to note that the definition of the Logical Schema is done in UML (Unified Modelling Language). The choice of this modelling language has been done according to the nature of the Oracle DBMS. Some additional information will be provided later:



Two main elements are defined in the schema, the **ORM User Type** representing the Types in Oracle DBMS and the **ORM Persistable** representing the Tables. The relationship between Types and Tables is defined using the aggregation with stereotype **Contained**.

PHYSICAL DESIGN

In the Physical Design phase, we are going to define the physical structures that able us to optimize the most frequent queries. There's not a standard process in this phase because depends strictly on the DBMS chose.

However, we can define the physical structures basing on the operation type and the frequency. An insert operation doesn't require an index because doesn't perform any search of scan. However, a search or scan operation that requires also joins between table, may be need an index on its fields.

Oracle defines for default indexes on the primary key of a table, representing an important advantage for us.

In the next lines we will define the operation, analysing if it's necessary define an index and the possible improve in the definition.

First, is important to note that Oracle able us the definition of two main physical structure, the B-tree and Bitmap. In addition, indexes are already defined on the primary key attribute of a table, thus the operations involving the primary key will not be optimized.

Operations number **1**, **2, 8** are insert operations so they don't require any indexes definition.

Operation number **3** requires the finding of all unbranded drugs equivalent to a branded drug sorted by name. We could optimize the query defining an index on the column "Name" of the table "UnbrandedMedicines", however, how said before Oracle defines already indexes on the primary key of a table, which is in this case the attribute "Name".

Operation number **4** requires printing all the drugs sorted by the number of prescriptions in which they are present. Because of, to find the Medicines of a Prescription we use the primary key of the Prescriptions table, we don't need to use an index to optimize the query, because it's already defined in Oracle.

Operation number **5** requires printing all the Prescriptions for a given doctor. However, the relation between Doctor and Prescription is realized using a pointer, making a direct reference between the two entities, so it's no necessary to define an index for this operation.

Operation number **6** requires finding all the Sales sold in a specific time range giving in input. For this query, we can improve the performances defining partition based on the year of the Sales. In this way, we reduce the rage interval scan for each query to a reasonable number.

Operation number **7** requires the authentication of the User based on the credential and the password. We can optimize the query defining an index over the field Nickname.

Operation number **9** requires performing the statistics associated to the Reports. Supposing that the Reports are calculated per year, we can define a partition on this field in order to decrease the workload.

TRIGGER DESIGN, VIEWS AND FUNCTIONS

Based on the Constraint defined using the Business Rules, we have to define some Active Rules (called Trigger) that able us to satisfy the constraints imposed by them. Following, for each Business Rule is reported the Trigger's name that satisfy it and the event that trig it.

Business Rule Code	Trigger Name	Event
C_2	CheckLicences	Before insert on Branded Medicines Before Update of Year field on Branded Medicines
C_3	CheckReportYear	Before Insert on Reports Before Update of Year on Reports
C_4	CheckPerfumeryNames	Before Insert on PerfumeryProducts Before Update of Type on PerfumeryProducts.
C_5	CheckMedicinesPrescription	Before Insert or Update on Prescriptions.
C_6	CheckSaleDate	Before Insert on Sales Before Update of Day on Sales.

Note that the Business Rule 1 is already satisfied by the nature of the Logical Schema. In addition, other rules are defined, to manage the constraint imposed naturally by the Conceptual Schema but not directly expressed in the Logical Schema. Some rules have been defined to make easier further operations.

Trigger's name	Description	Event
CheckSaleProductName	Check if the product name insert exists or not.	Before Insert on Sales Before Update on Sales
CheckUsers	Check if the current user is already registered to the platform	Before Insert on Users
CheckNikname	Check if the insert Nickname is already used by another User	Before Insert on Users Before Update of Nickname on Users.

Maybe, the most important trigger is **UpdateReports**, in fact the Reports related to a Sale are generate automatically by the trigger each time a new Sale is updated, deleted or inserted into the Database. We will explain the trigger's behaviour in terms of the events that trigger it.

- **Inserting**, examining the data of the new Sale, the values of the semester and year of the Report are generate, if a Report with the following semester and year is present in the Database, the amount of the report is increased and the Sale is linked to it using the REF.
- **Updating**, only on the Day field of Sale. The new Report year and semester are generated and if exists a Reports with the following information, the Sale is linked to this new. Then, the old Report amount is decreased by the total of the Sale, and the new Report's amount is increased by the total of the Sale.
- **Deleting**, the Report's amount linked to the Sale is decreased by the total of the Sale.

To make easier some operations raising the Application to performing complex queries, have been decided to use some Views, also for the following reasons:

- Simplifying data retrieval, raising the Application to perform complex queries.
- Maintaining logical data independence, exposing to the Application only the needed information.
- Implementing data security, adding an additional security layer.

The Views have been defined based on the operations exposed by the Database, following we can map the operations with the Views:

Operation's code	View's name
3	UnbrandeMedicineEquivalent
4	PrintMedicine
5	GenerateList
6	SalesRange
7	RegisteredUser
9	ReportsStatistics

To make easier the performing of some functionalities, from the Server's side without the explication of long queries, some functions have been created. Following, we will map the operation's identified in the specification with the functions defined in the Database's schema.

Operation Code	Function's name	Description
7	InsertNewUse	Register the user's specified in input in the database, with the crypted password.
1	InsertNewMedicine	Register the medicine in input in the Database.
1	InsertNewMedicineWithCompany	Register the new medicine in input and the Company.
1	InsertCosmetic	Register a new Cosmetic in the Database,

	InsertNewSale	Register a new sale in the Database with the first product associated.
	RegisterNewSale	Add a product to an existing sale.
8	InsertNewPrescription	Register a new prescription with the first associated medicine.
8	RegisterNewProduct	Add a new medicine to an existing prescription.

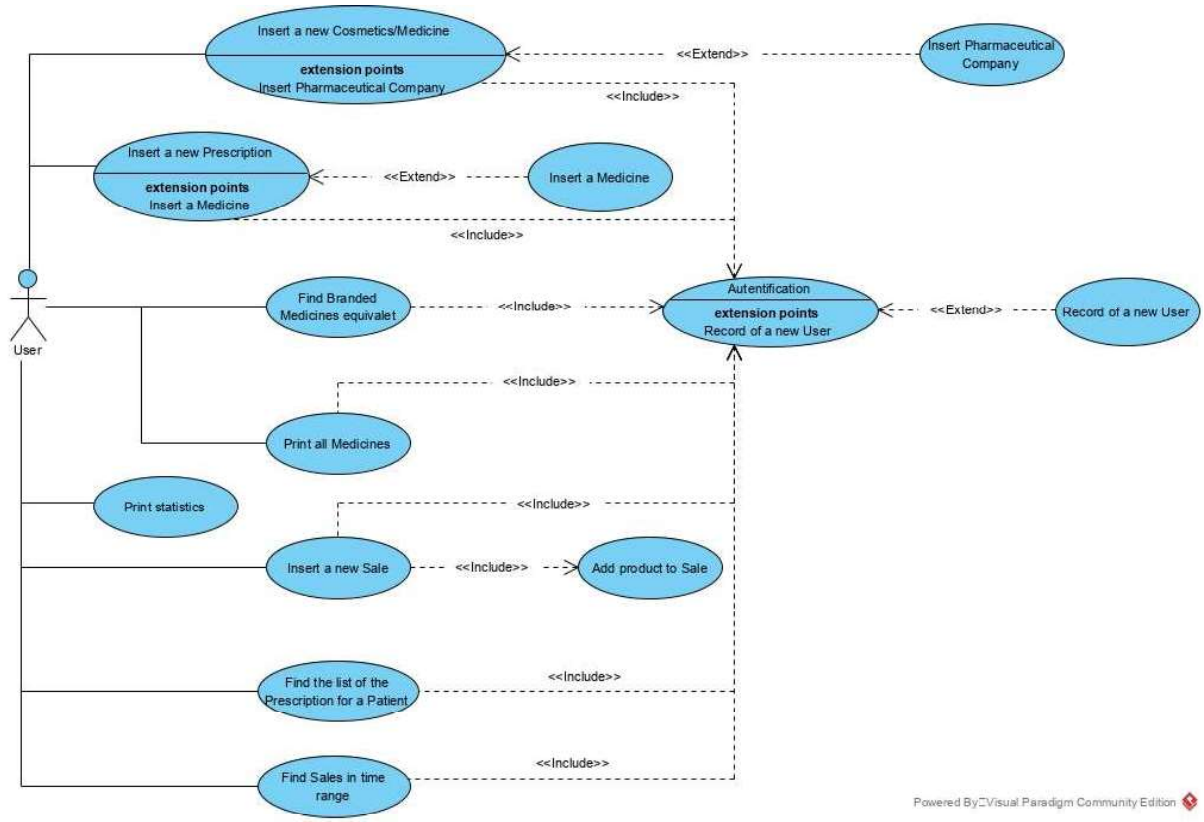
APPLICATION DESIGN

The second phase of the project is the design of the Application which interfaces with the Database according to the defined Architecture. This section in articulated as follow: the first part is concerned on the functional requirements that the Application must have; following will be refined the architecture defined previously, defining the physical components of the Application and at the end the implementation of each functionalities associated to the identified components.

As mentioned in the beginning phase, the application will be designed using the Java Enterprise Edition and the Java Server Faces to generate to HTML code. The HTML pages will use Javascript and AJAX to communicate with the HTTP protocol with the Application, which will use the JDBC interface to communicate with the Oracle Database.

FUNCTIONAL REQUIREMENTS OF THE APPLICATION

The Functional Requirements of the Application are expressed under the form of Use Cases, for each of these reported in the diagram will be provided a description in tabular form, to get an idea of the interaction between the User and the Application and the work to perform.



Use Case's Name	Authentication
Primary Actor	User
Secondary Actor	System
Pre-conditions	None
Post-conditions	The user is identified successfully
Scenario's description	1. The user connects to the Application. 2. The user types his credentials. 3. The system identifies the User.
Alternative Scenario	3.a The user is not identified. 3.b The system responses with an error message.
Extensions	Record a new User

Use Case's Name	Record a new User
Primary Actor	User
Secondary Actor	System
Pre-conditions	None
Post-conditions	The user is recorded successfully
Scenario's description	1. The user connects to the Application. 2. The user completes the registration's form. 3. The system records the User.
Alternative Scenario	3.a The user is already recorded. 3.b The system responses with an error message.
Extensions	None

Use Case's Name	Insert a new Cosmetic/Medicine
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	The Cosmetic/Medicine is inserted successfully.
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user completes the form for insert the Cosmetic/Medicine.
Alternative Scenario	3.a The Cosmetic/Medicine is already recorded. 3.b The system responses with an error message.
Extensions	Insert pharmaceutical company.

Use Case's Name	Insert a new Prescription
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	The Prescription is inserted successfully.
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user completes the form for insert the Prescription.
Alternative Scenario	3.a The Prescription is already recorded. 3.b The system responses with an error message.
Extensions	Insert a Medicine.

Use Case's Name	Find Branded equivalent Medicine
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	None
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user types the Medicine Branded name that want to find. 4. The system shows the result records
Alternative Scenario	4.a There are no Branded Medicine with that name. 4.b The system responses with an error message.
Extensions	None

Use Case's Name	Print all Medicines
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	None
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user types the Medicine name that want to find. 4. The system shows the result records
Alternative Scenario	4.a There are no Medicine with that name. 4.b The system responses with an error message.
Extensions	None

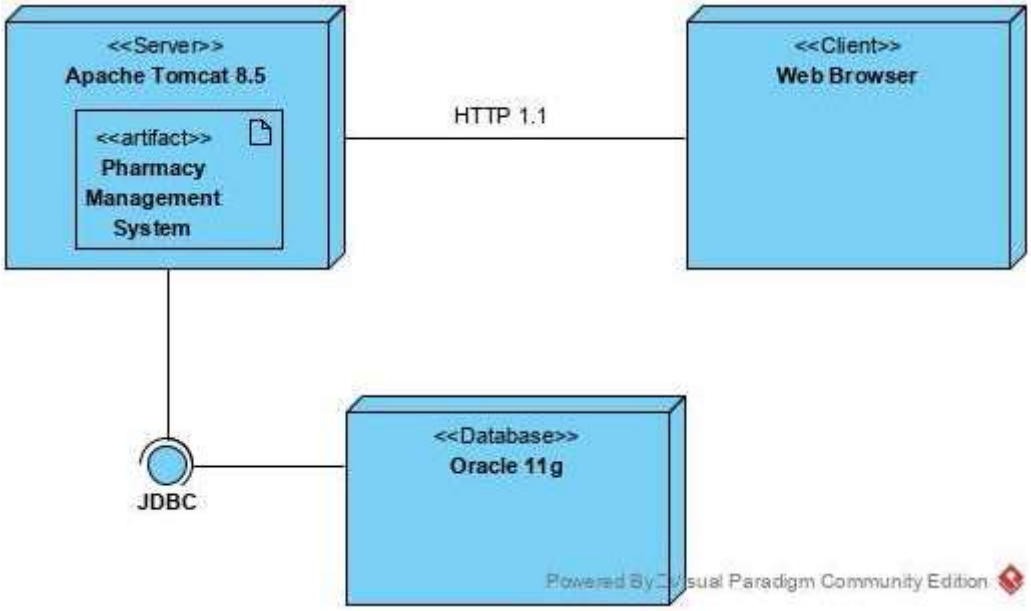
Use Case's Name	Insert a new Sale
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	The Sale is recorded successfully.
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user completes the form for insert a new Sale. 4. Include **Add Product to Sale.**
Alternative Scenario	None
Extensions	None

Use Case's Name	Find the list of Prescriptions of a Patient
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	None
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user types the Patient's name. 4. The system shows the result records.
Alternative Scenario	4.a There are no result records in the Database. 4.b The system responses with an error message.
Extensions	None

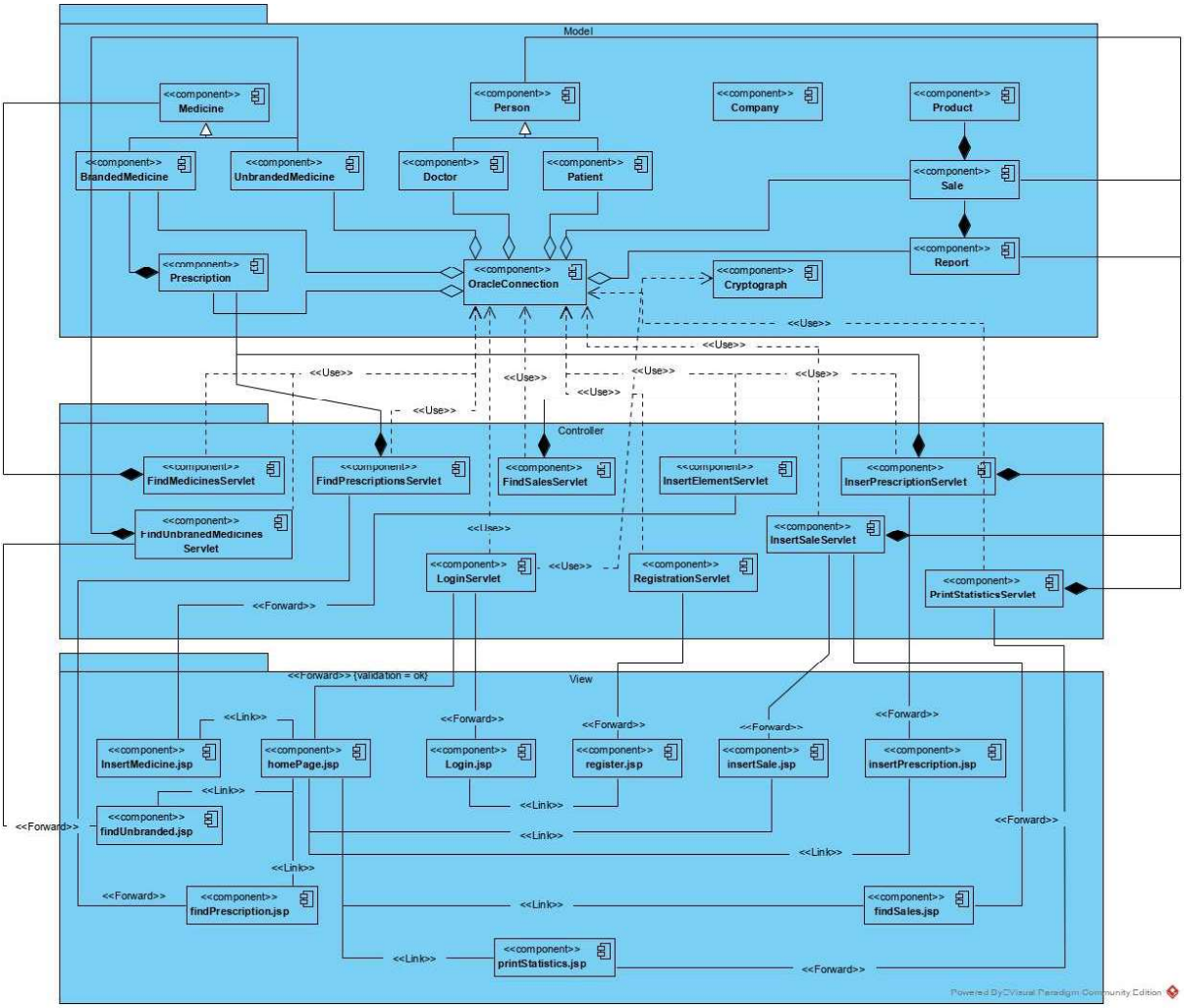
Use Case's Name	Find Sales in the range
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	None
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The user types the Sale's date range to show. 4. The system shows the result records.
Alternative Scenario	4.a There are no result records in the Database. 4.b The system responses with an error message.
Extensions	None

Use Case's Name	Print statistics
Primary Actor	User
Secondary Actor	System
Pre-conditions	The user must be authenticated.
Post-conditions	None
Scenario's description	1. The user connects to the Application. 2. Include **Authentication**. 3. The system shows the statistics associated to the Sales.
Alternative Scenario	3.a There are no result records in the Database. 3.b The system responses with an error message.
Extensions	None

Following, after the definition of the Functional Requirements that the application must have, we can proceed defining the distribution of the Software and the components in it.



According to the refined architecture, the application, whose name is Pharmacy Management System, internal distribution of components is the following.



The Application have been designed following the Architectural Pattern Model-View-Controller, having a clear separation between the duty of different components.

The Controller component contains the Java Servlet which deals with to be a Dispatcher between the HTTP requests made by the JSP pages and the Oracle Database, using the component *OracleConnection* defined in the Model component. Each Servlet is distinguished by the other using a unique identifier, a HTTP request method, which can be GET or POST only (others methods like PUT or DELETE are not used anymore), and URL pattern. Following, we will describe each Servlet based on these elements:

- *LoginServlet*, operated on the URL pattern *"/login"* and response to an HTTP POST request coming from the JSP page *"login.jsp"*. The behaviour is this: the first page showed to the user is the *"login.jsp"*, this page contains a form for typing the credential with perform the POST request to the Servlet. After receiving the request, the Servlet uses the OracleConnection Interface to check the credential and

response with an OK or Error message to the page, indicating if the credentials are valid or no.

- *RegistrationServlet*, operated on the URL pattern "/register" and response to an HTTP POST request coming from the JSP page "register.jsp". The behaviour of this is the following: the page "*register.jsp"* perform an HTTP POST request to the servlet containing the form's parameters. After receiving the request, the Servlet uses the OracleConnection interface, the Cryptographer Class and the DateUtilities class to manage the parameters and query the database. If the registration goes well the Servlet will respond with an Ok message else an Error message.
- *FindMedicineServlet*, operate on the URL pattern "/findMedicine" and response to the HTTP GET and POST requests from the JSP "findMedicines.jsp". The Servlet find all the branded medicines in the database and send the result to the JSP page, then when a POST request is received, delete from the database the chosen element.
- *FindPrescriptionServlet*, operate on the URL pattern "/printPrescriptions" to the HTTP GET and POST requests from the JSP "findPrescription.jsp". The Servlet return to the page all the prescriptions in the Database and delete those sanded through the POST request.
- *FindSalesServlet*, operate on the URL pattern "/findSales" and response to the HTTP GET and POST requests from the JSP "findSales.jsp". The servlet returns to the page all the sales through the GET request, if the user specify a particular time range in input, the Servlet will return the corresponding elements. In addition, the Servlet delete from the database the corresponding element in input.
- *FindUnbrandedMedicines*, operates on the URL pattern "/findUnbranded" and respond to the HTTP GET and POST methods, retrieving all the unbranded medicines in the database and deleting those specified in input from the page "findUnbranded.jsp" through the POST method.
- *InsertElementServlet*, operates on the URL pattern "/insertElement" and response to the HTTP GET and POST requests from the JSP page "insertMedicine.jsp". Through the GET request, the Servlet response sending all the Names of Companies in the Database. The HTTP POST request manage the insertion of a new medicine/cosmetic.
- *InsertPrescripionServlet,* operates on the URL pattern "/insertPrescription" and response to the HTTP POST request inserting the specified Sale in input from the page "insertPrescription.jsp".
- *InsertSaleServlet,* operates on the URL pattern "/insertSale" and response to the HTTP POST request inserting the specified Sale in input from the page "insertSalejsp".

- *PrintStatisticsServlet*, operated on the URL pattern "/printStatistics" response to the HTTP GET request, retrieving all the reports in the database.

Each servlet uses one or more Class defined into the Model package, following we will give a description of these classes and the Servlet that makes use of these.

Class' name	Class' description	Servlet
Medicine	Abstract class that define a generic medicine.	FindMedicineServlet
BrandedMedicine	Concrete class that defines a branded medicine.	
UnbrandedMedicine	Concrete class that defines an unbranded medicine.	FindUnbrandedMedicineServlet
Person	Abstract class that defines a generic person	
Patient	Concrete class representing a patient.	
Doctor	Concrete class representing a doctor.	InsertPrescriptionServlet
Company	Concrete class representing a company that produces a medicine.	
Product	Concrete class representing a sold Product.	
Sale	Concrete class representing a generic Sale,	InsertSaleServlet

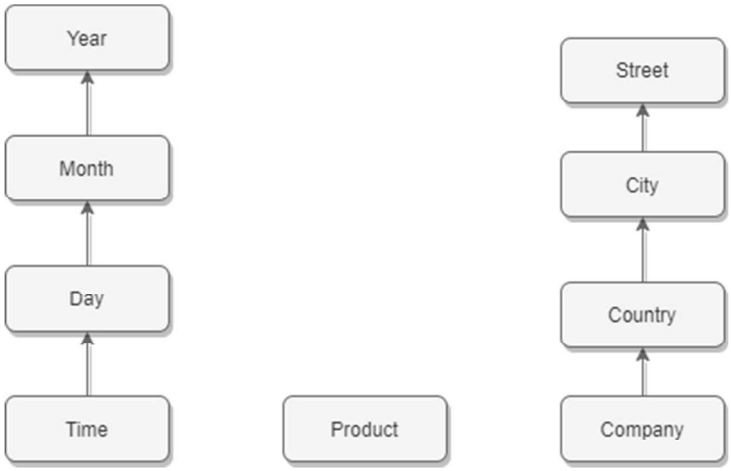
	containing some products.	
Report	Concrete class representing a Report, containing some sales.	PrintStatisticsServlet
Cryptograph	Concrete class used to encrypt and decrypt the password of an user	LoginServlet
OracleConnection	Main class used to model the connection with the database. Designed based on the Design Pattern Singleton.	All servlets.

DATAWAREHOUSE DESIGN

Allowing to perform some OLAP operations by the pharmacy admin, a Datawarehouse schema have been designed starting from the Conceptual schema of the Database. Following, we will identify the facts, measures and dimensions.

The fact represented in the schema is the Sale of the products in the Database, the dimensions are the Products and Time and Company.

For the identified dimensions it's possible to identify the following hierarchy:



Based on these information, we can define the following star schema: