Health System Application

Analysis and Design Document

Student: Bucur Diana

**Group: 30433**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/04/2018 | 1.0 | Iteration 1 | Bucur Diana |
| 27/05/2018 | 2.0 | Iteration 2 | Bucur Diana |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

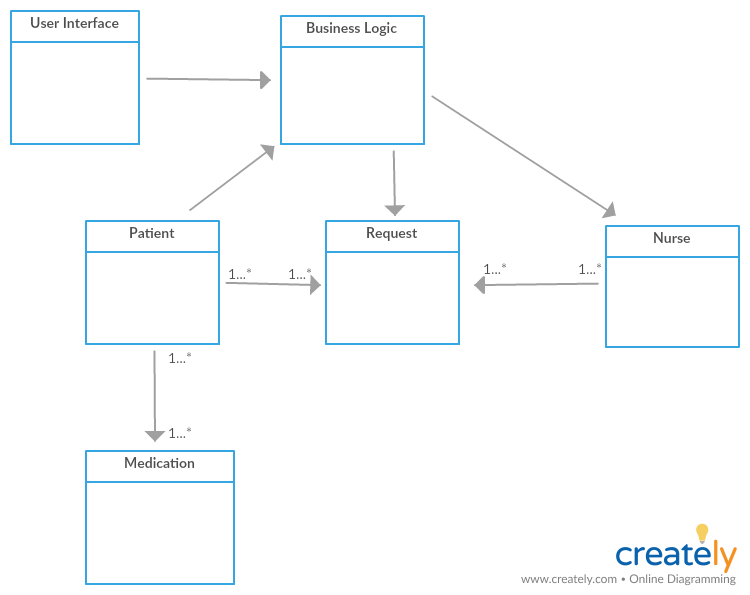
VI. Bibliography 5

# Project Specification

*[Present the project specification]*

# Elaboration – Iteration 1.1

# Domain Model



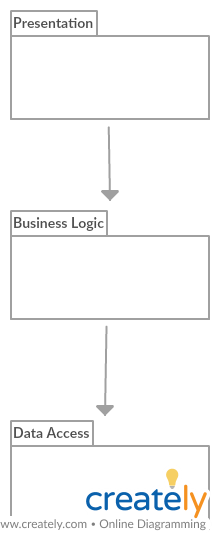
# Architectural Design

## Conceptual Architecture

The system will use a Client-Server architecture, which is an architecture of a [computer](https://www.britannica.com/technology/computer) [network](https://www.britannica.com/technology/computer-network) in which many [clients](https://www.britannica.com/technology/client) (remote processors) request and receive service from a centralized [server](https://www.britannica.com/technology/server) (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. These fits the requirements of the system, as the nurses are supposed to receive requests from patients and respond to them.

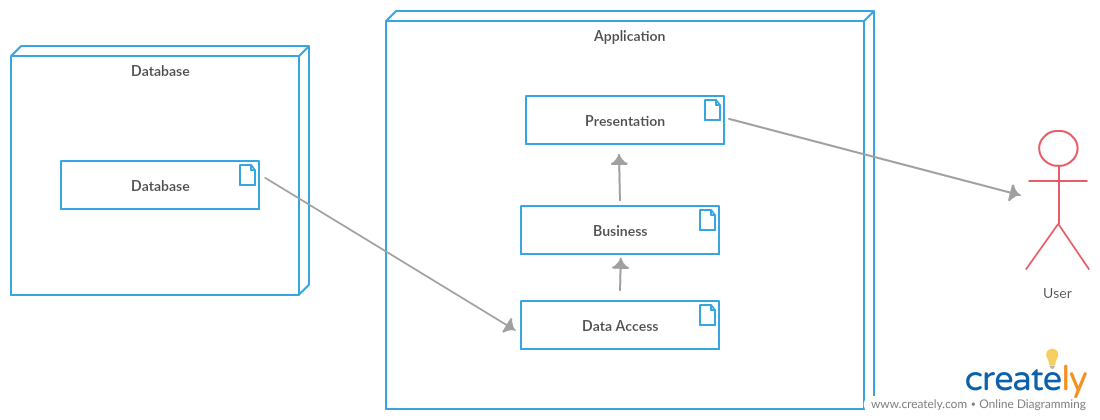
This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform. In [hospital](https://www.britannica.com/science/hospital) [data processing](https://www.britannica.com/technology/data-processing), for example, a client computer can be running an application program for entering patient information while the server computer is running another program that manages the [database](https://www.britannica.com/technology/database) in which the information is permanently stored. Many clients can access the server’s information simultaneously, and, at the same time, a client computer can perform other tasks.

## Package Design



## Component and Deployment Diagrams

# Deployment diagram:

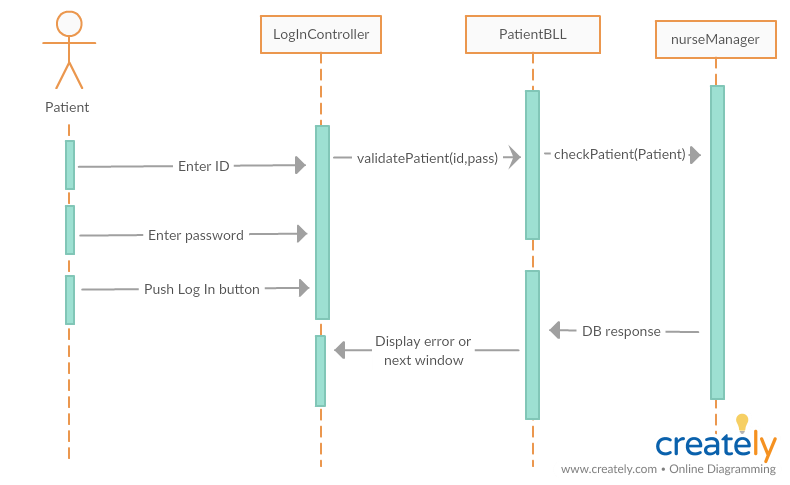


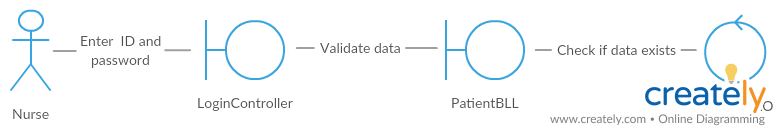
# Elaboration – Iteration 1.2

# Design Model

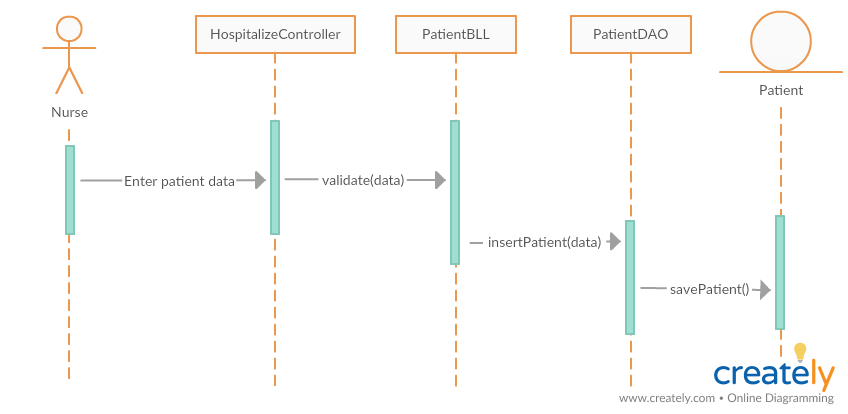
## Dynamic Behavior

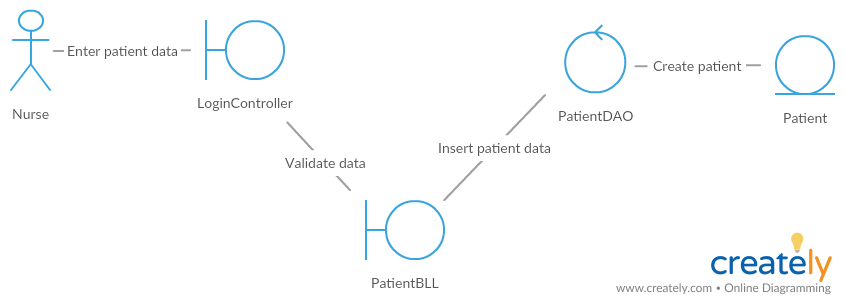
1.1.1 Log In Scenario



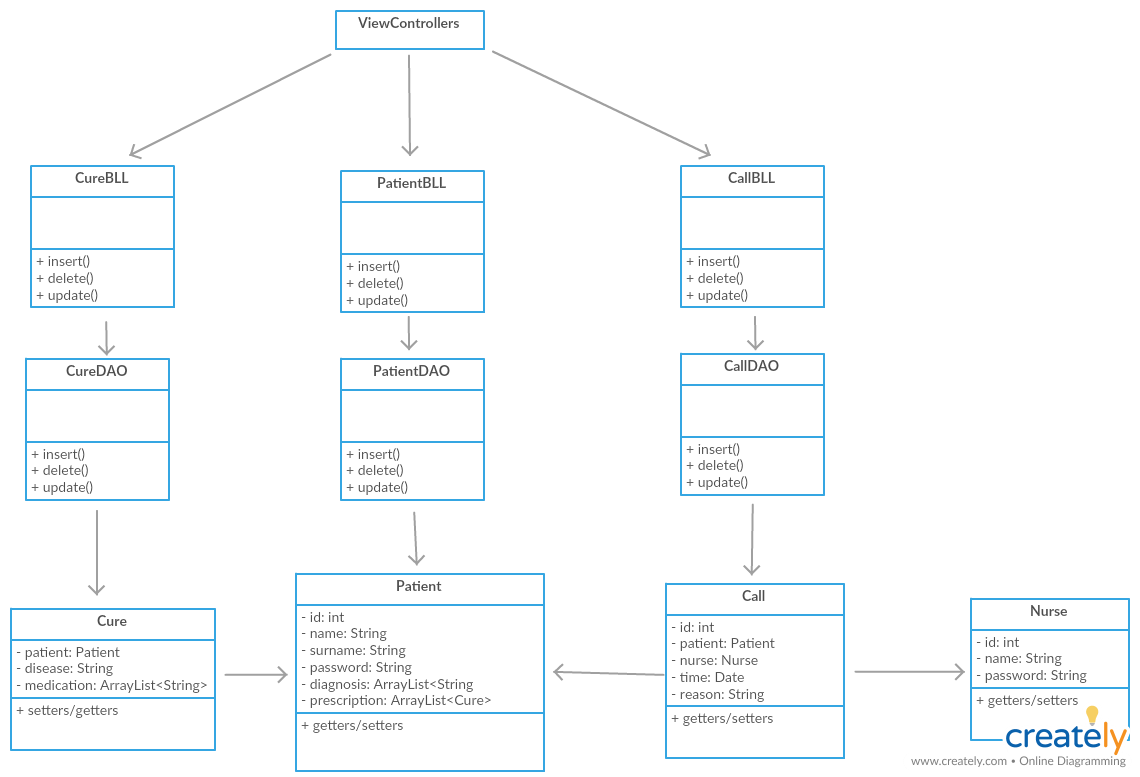


1.1.2 Hospitalize (create) patient Scenario

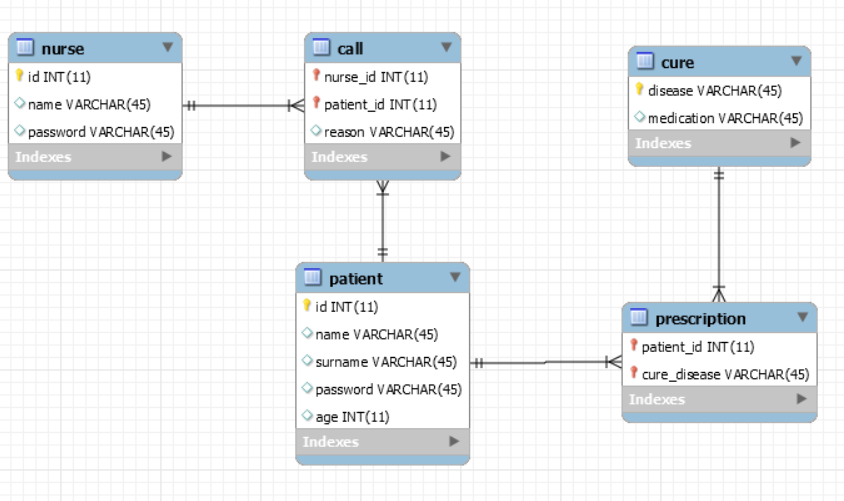




## Class Design



# Data Model.



# Unit Testing

Taking into consideration the future functionalities of the application, some unit tests would be useful to be implemented regarding the following situations:

- correctly retrieving information from the database

- verify if the algorithm assigning nurses to take the call functions properly

- check if all the nurses receive the requests on time

- check that the status of nurses is updated accordingly

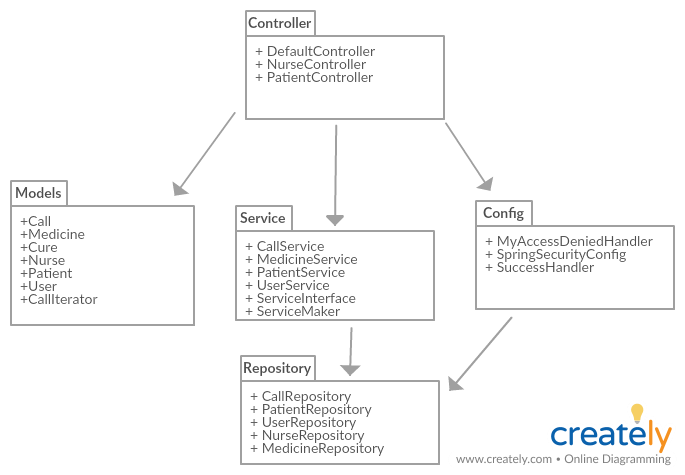
# Elaboration – Iteration 2

# Architectural Design Refinement

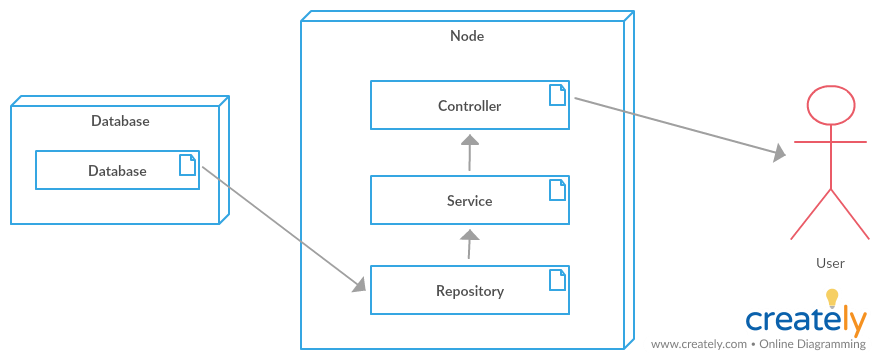
The Spring Framework is an [application framework](https://en.wikipedia.org/wiki/Application_framework) and [inversion of control](https://en.wikipedia.org/wiki/Inversion_of_control) [container](https://en.wikipedia.org/wiki/Servlet_container) for the [Java platform](https://en.wikipedia.org/wiki/Java_platform). The Spring Framework includes several modules that provide a range of services:

* [Authentication](https://en.wikipedia.org/wiki/Authentication) and [authorization](https://en.wikipedia.org/wiki/Authorization): configurable security processes that support a range of standards, protocols, tools and practices via the [Spring Security](https://en.wikipedia.org/wiki/Spring_Security) sub-project
* Data access: working with relational database management systems on the Java platform using Java Database Connectivity (JDBC) and object-relational mapping tools and with NoSQL databases
* Model–view–controller: an HTTP- and servlet-based framework providing hooks for extension and customization for web applications and RESTful (representational state transfer) Web services.
* Remote access framework: configurative remote procedure call (RPC)-style marshalling of Java objects over networks supporting Java remote method invocation and HTTP-based protocols including Web services (SOAP (Simple Object Access Protocol))

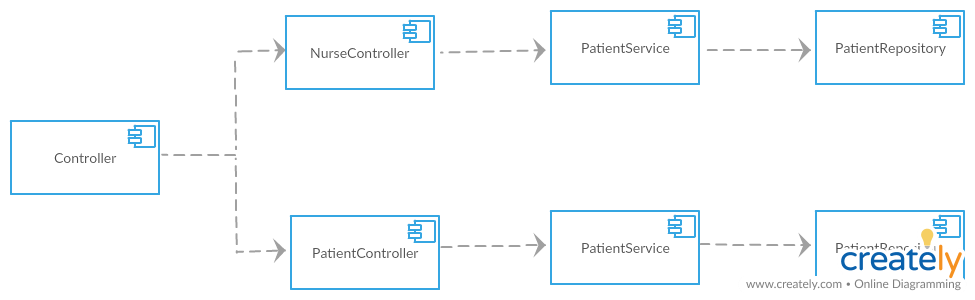
**1.1 Package Diagram**



**1.2 Deployment Diagram**



**1.3 Component Diagram**



# C:\Users\diana\AppData\Local\Microsoft\Windows\INetCache\Content.Word\sequencediag.pngDesign Model Refinement

# Construction and Transition

# System Testing

For system testing, I tried implementing a Junit test which verified that a patient is successfully hospitalized, meaning their information was registered in the database and can be retrieved at any moment.

# Future improvements

As far as future improvements are concerned, there are a lot of features that can be added to the current version of the application. First of all, it should be done more dynamic with the help of web sockets, in order for the view requests page for the nurses to be automatically updated in real-time when a patient makes a call.

Moreover, patients could make more personalized requests. As the possibility of seeing which nurse responded to a call would be implemented, patients could leave feedbacks for the assistance received. This way, the hospital employees would be more motivated to perform better their job and be penalized otherwise.

The user interface could be also improved with more css and html in order to make the application more appealing to both the patients and nurses and make it easier to use.

# Bibliography

1. <https://github.com/buzea/SoftwareDesign2018>
2. <http://www.mkyong.com/tutorials/spring-boot-tutorials/>
3. <https://www.thymeleaf.org/doc/tutorials/2.1/thymeleafspring.html>
4. <https://www.journaldev.com/3793/hibernate-tutorial>