Internship in Sight

Analysis and Design Document

Student: Cârcu Bogdan

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/04/2018 | 1.0 | Project Deliverable (2) | Cârcu Bogdan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

**Internship in Sight** is a Java-based application that targets the students and their need to find internships, training camps or other general work offers from companies. Also, companies will able to advertise their internship opportunities and interact in a transparent manner with the possible applicants. The application will support two main types of users: students and companies. Only one account per user and per company is permitted. Moreover, a third type of user will have total access to the overall information (the administrator).

The **administrator** can perform the following operations:

* CRUD on users
* CRUD on internships

The basic **user** can:

* Login
* Register
* Log out

The **student** can:

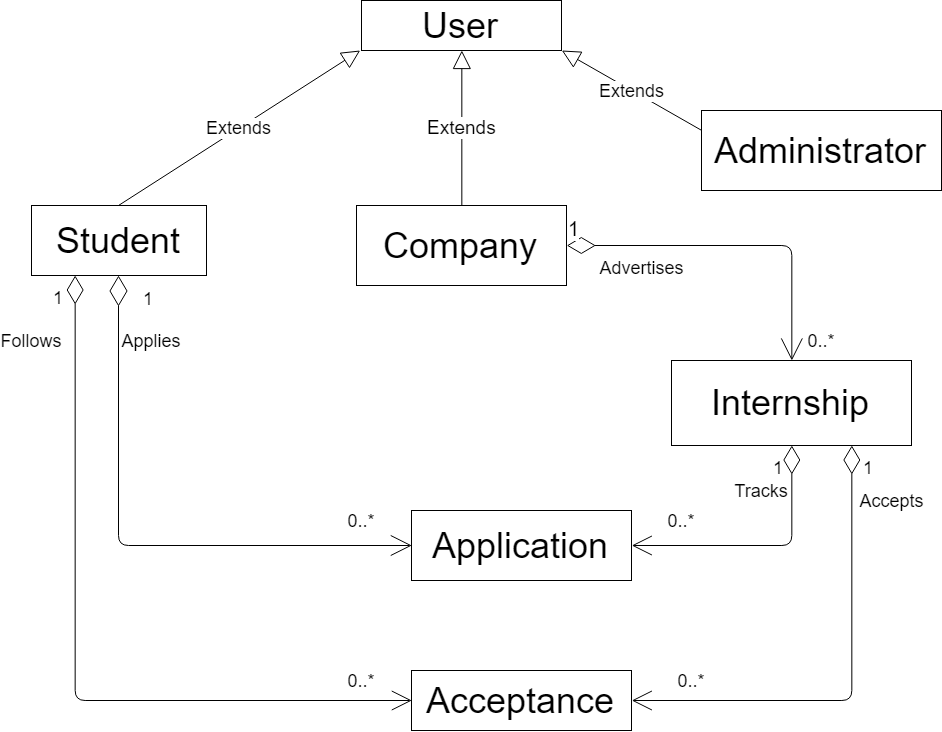
* View internship offers
* Apply for an internship
* Rate internship experience if participated in one
* Comment the internship experience

The **company** can:

* Accept a candidate
* Deny a candidate
* Advertise internships
* Tag students as their internship graduates, thus granting them access to the rating system

# Elaboration – Iteration 1.1

# Domain Model

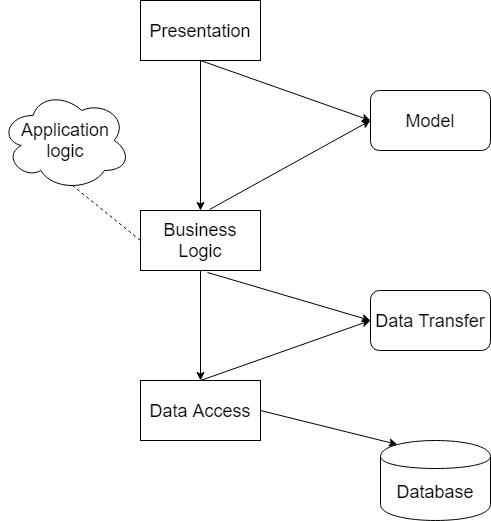


# Architectural Design

## Conceptual Architecture

Architectural style: **Layered Architecture**

* This architecture is a client-server architecture in which presentation, application processing, and data management functions are separated.
* Every layer can use the functionalities of the layer below but not vice-versa



**Motivation:**

* Coherent structure
* Ability to separate key enterprise functions into different logical locations where they can be executed, managed and changed with relative independence
* The platform trying to be built can be enhanced in a various number of ways, so we need support for frequent and unavoidable changes

Architectural Pattern: **Model-View-Controller**

**MVC** is an [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern) commonly used for developing [user interfaces](https://en.wikipedia.org/wiki/User_interface) that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient [code reuse](https://en.wikipedia.org/wiki/Code_reuse) and parallel development.

**Components**

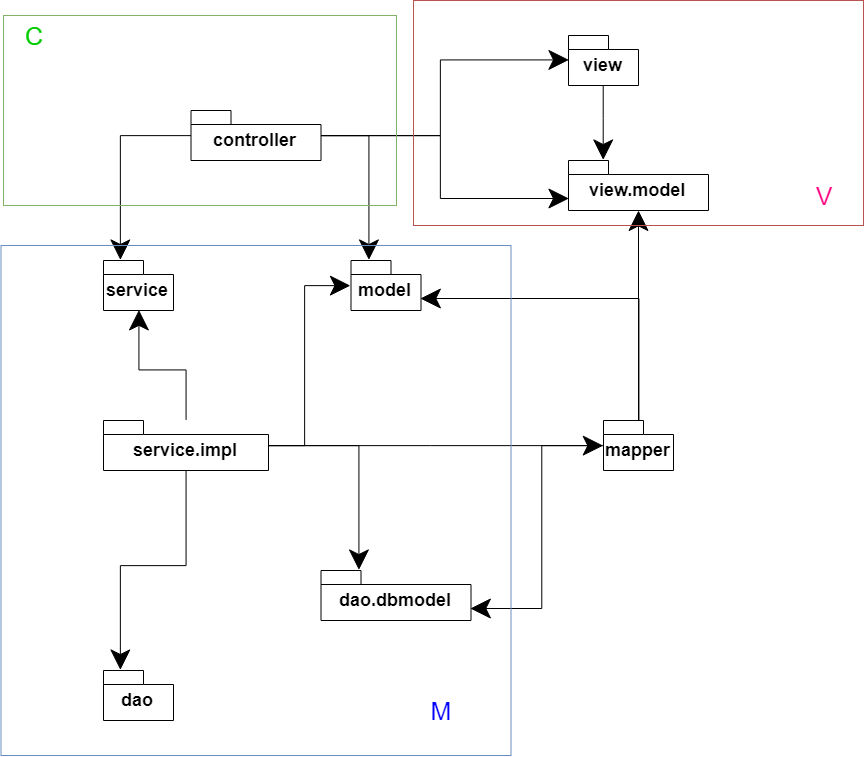
* The *model* is the central component of the pattern. It expresses the application's behavior in terms of the [problem domain](https://en.wikipedia.org/wiki/Problem_domain), independent of the user interface. It directly manages the data, logic and rules of the application.
* A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible.
* The third part or section, the *controller*, accepts input and converts it to commands for the model or view.



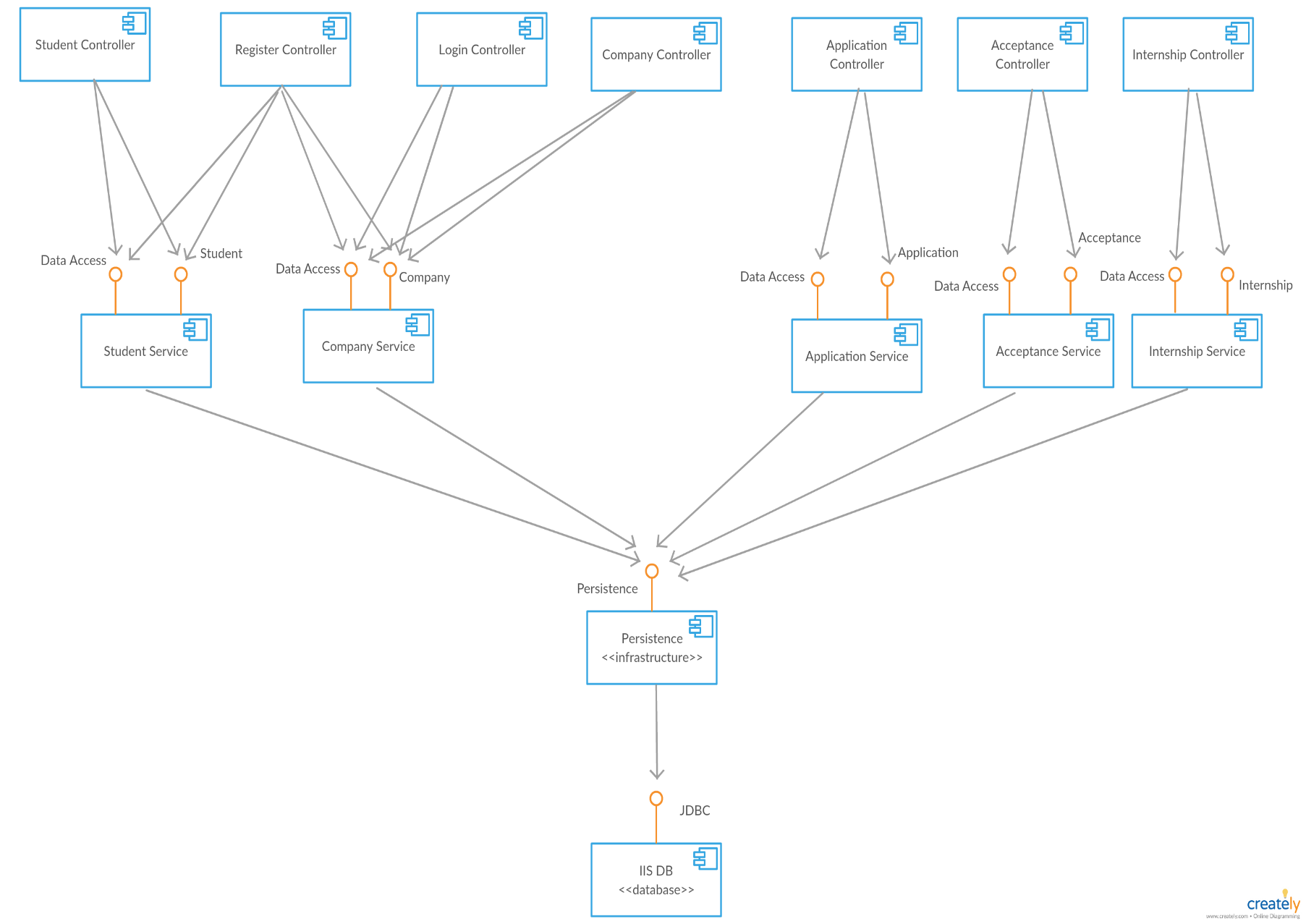
**Motivation:**

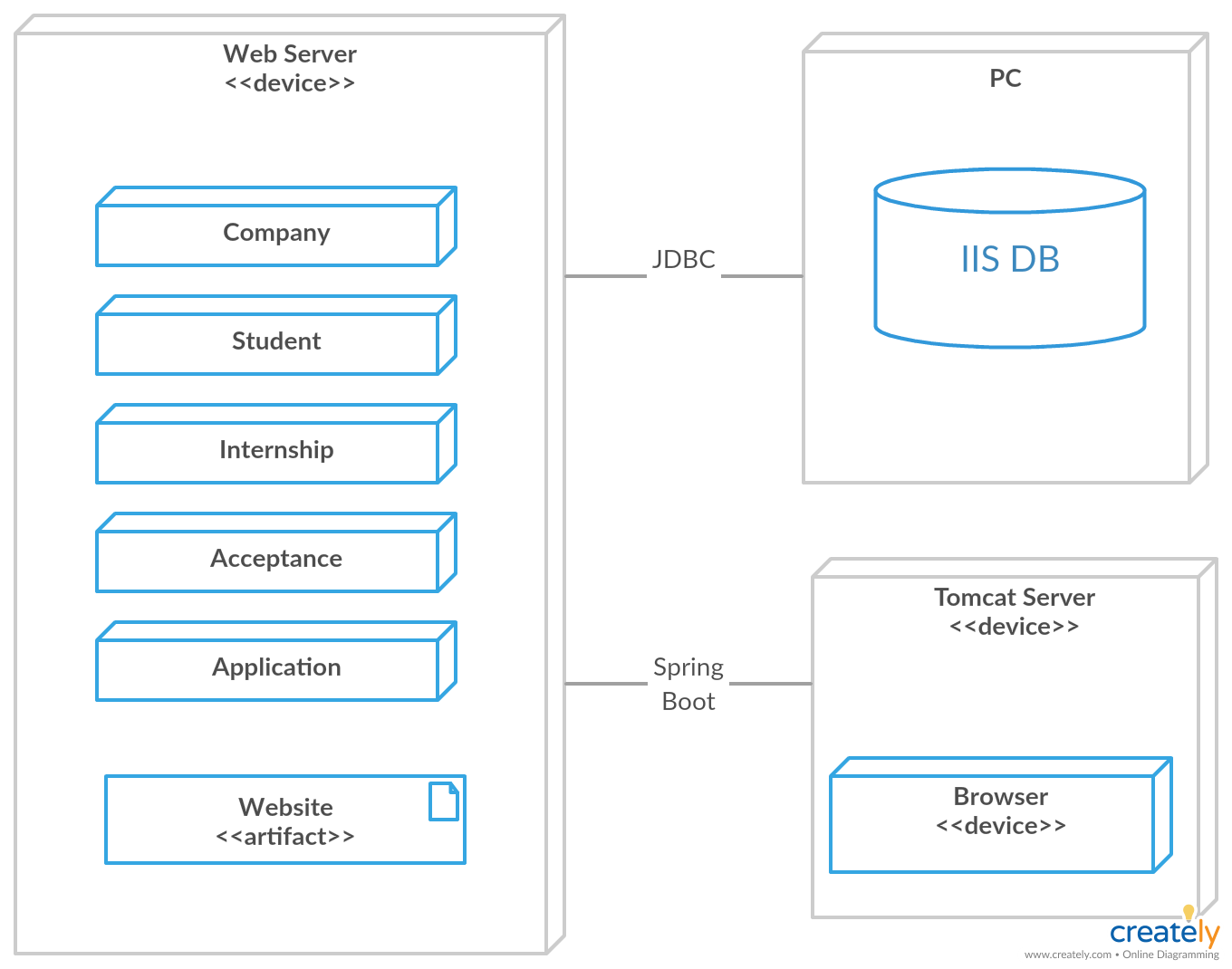
* Known to be one of the best choices when it comes to web-applications
* Flexible
* Adds bonuses to the separation of concerns
* Many platforms and frameworks provide clean and suggestive ways to implement a MVC system (e.g.: Spring Boot for Java)

## Package Design

**

## Component and Deployment Diagrams



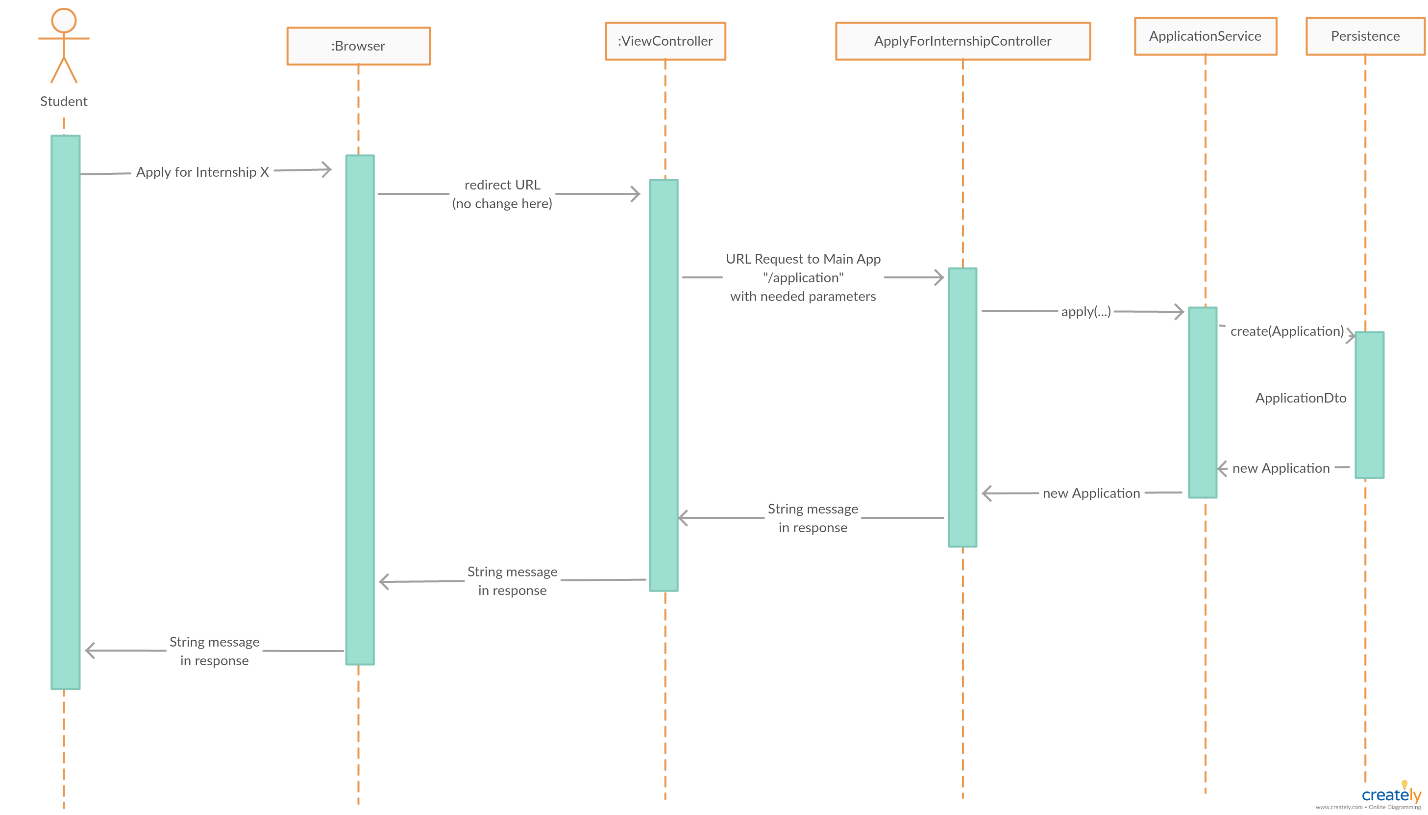


# Elaboration – Iteration 1.2

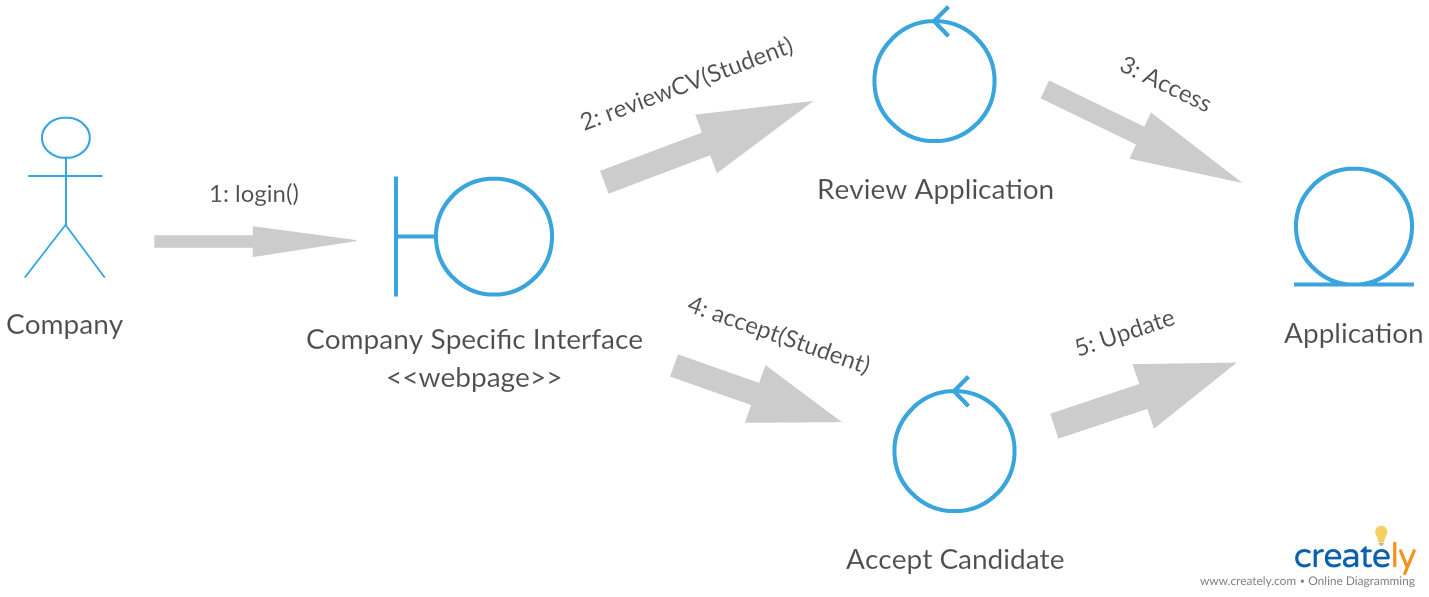
# Design Model

## Dynamic Behavior

* Student applying for internship X

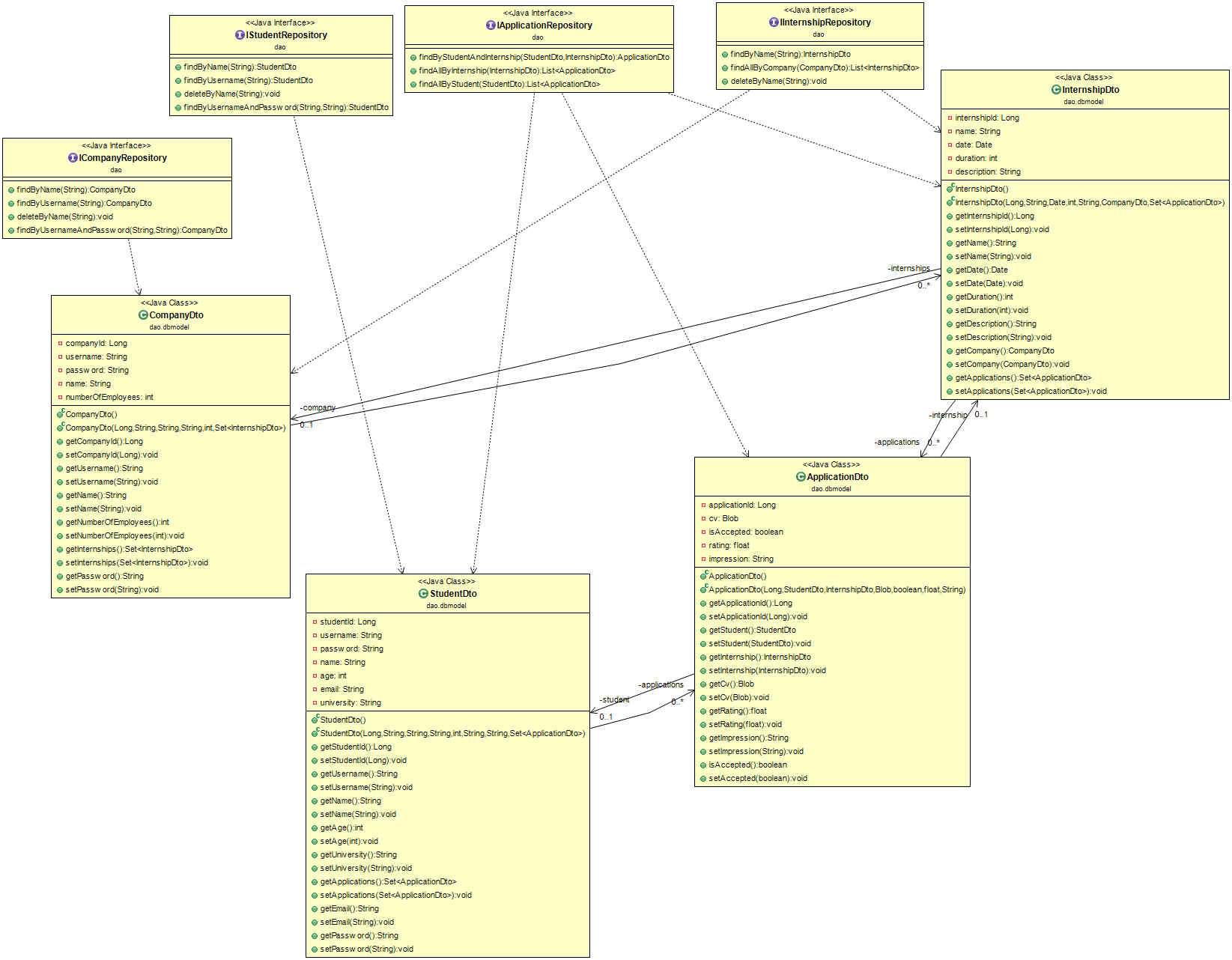
**

-Company reviews and accepts a candidate based on their application

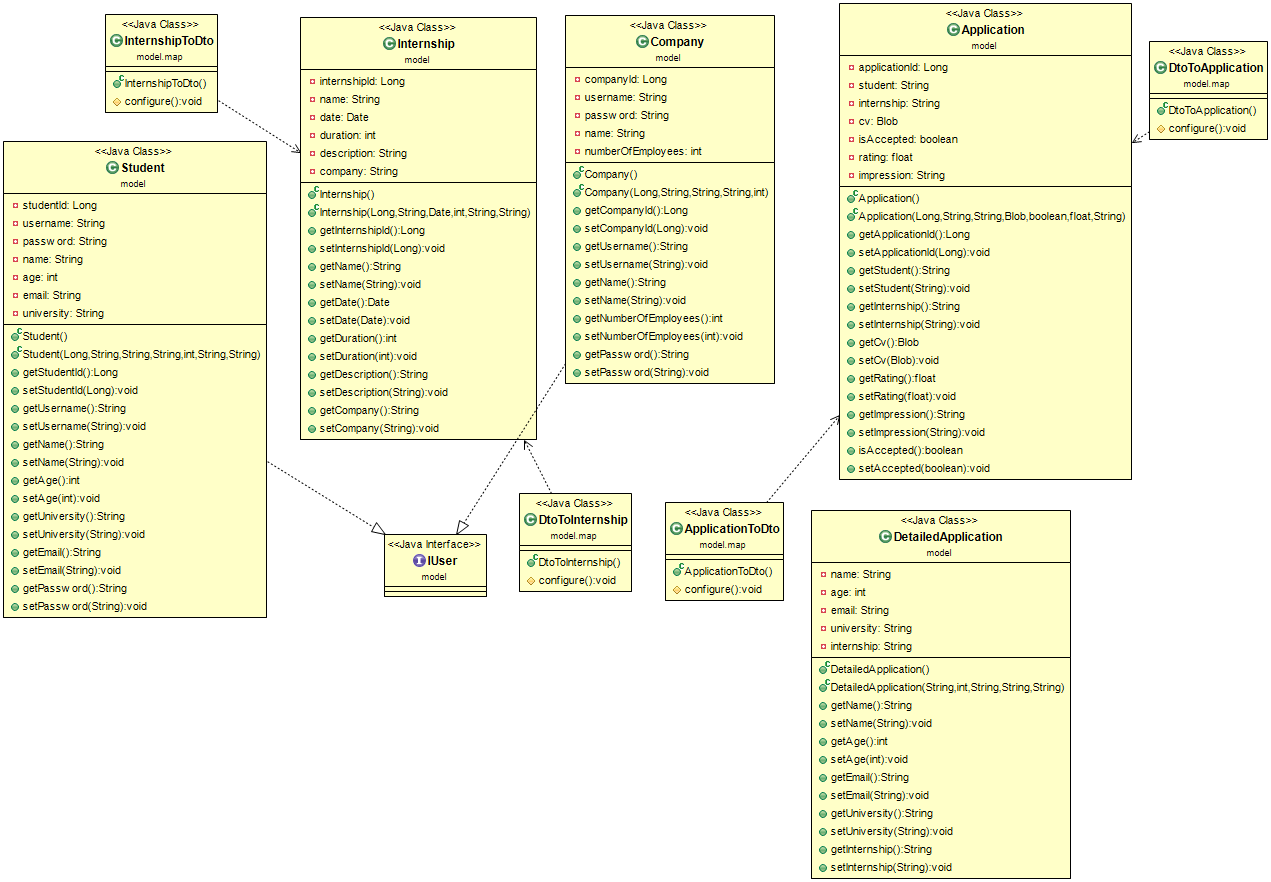
**

## Class Design

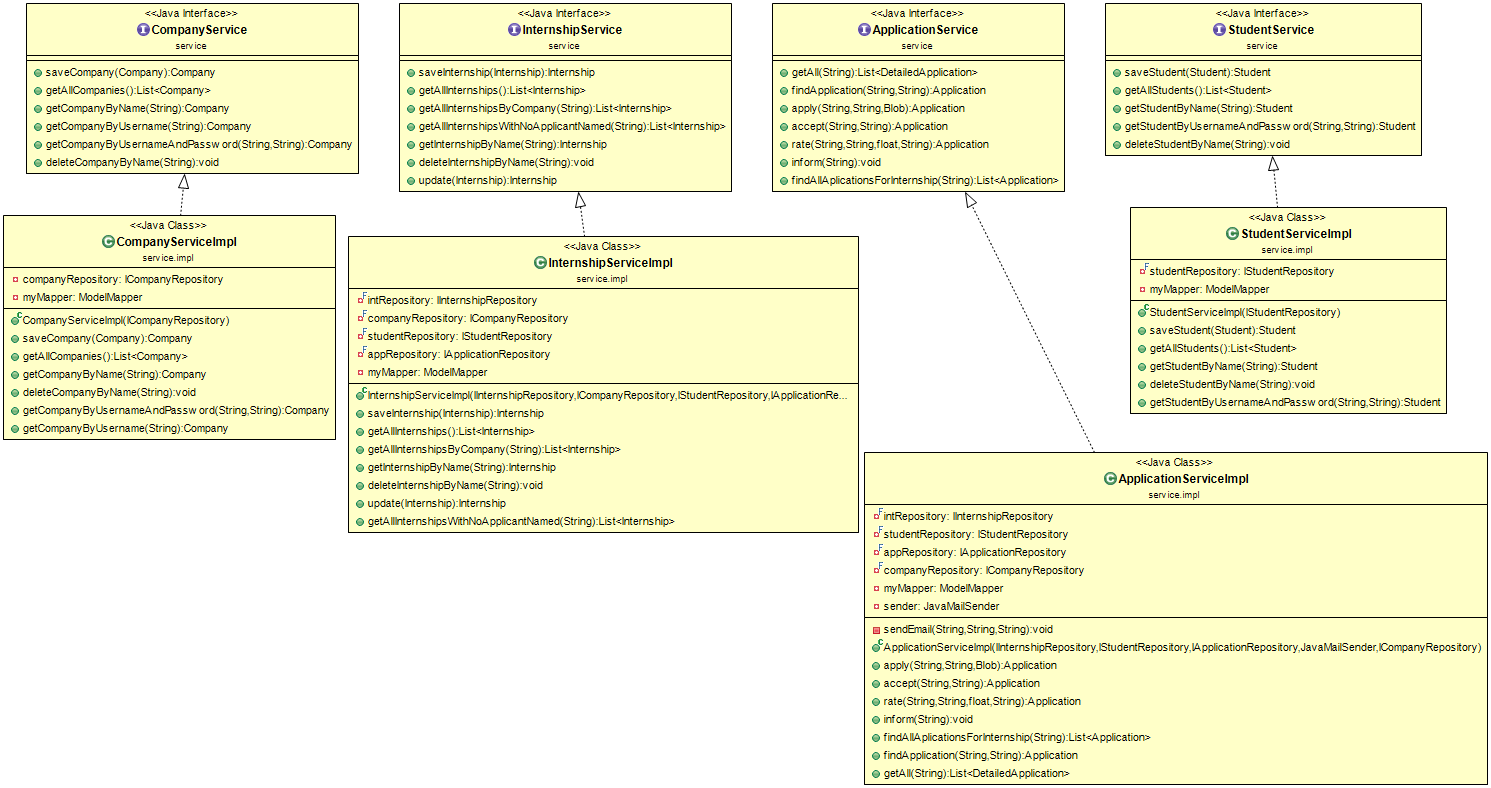
-Repositories



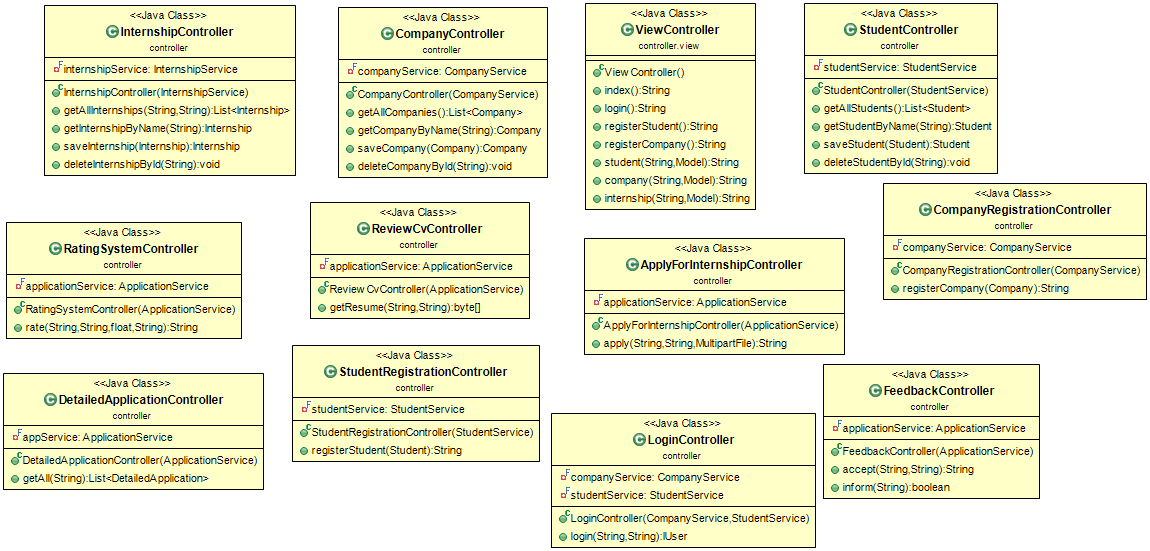
-Models



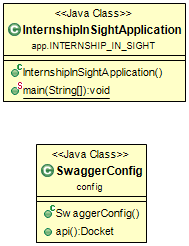
-Services



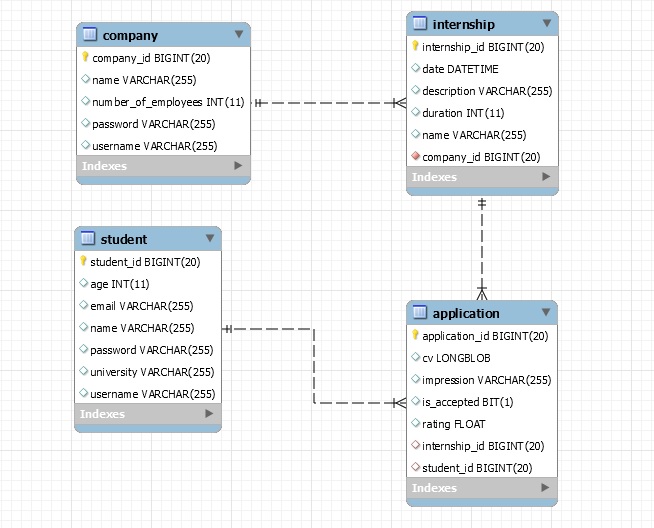
-Controllers



-Application

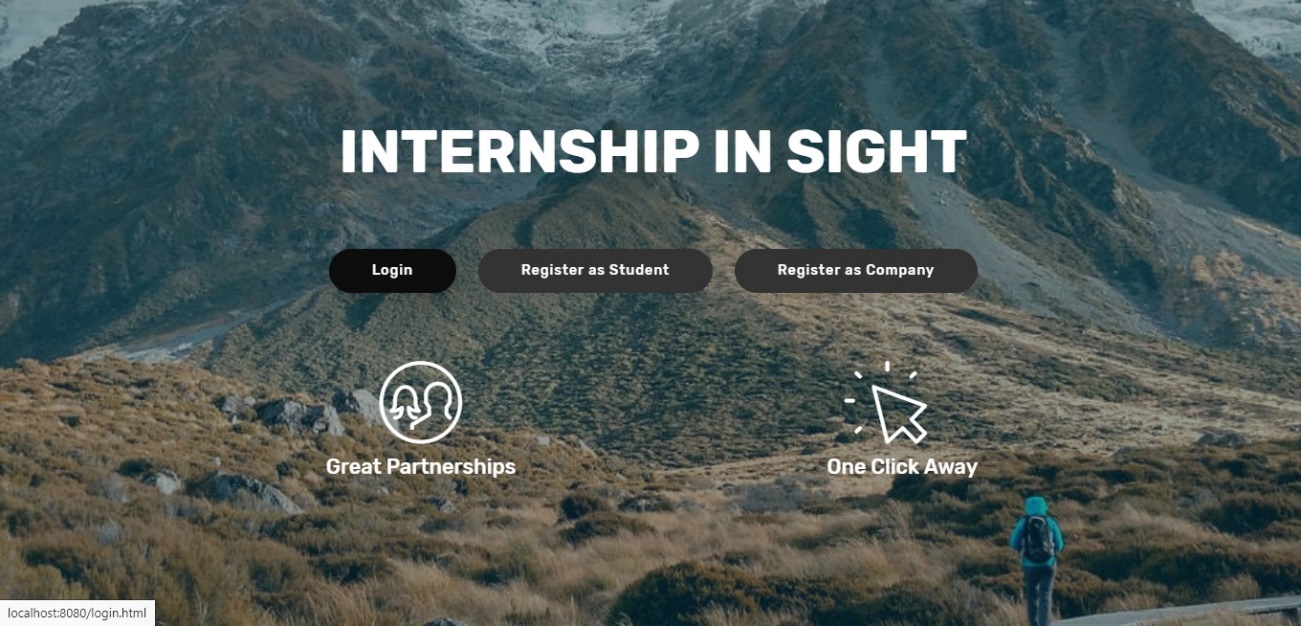


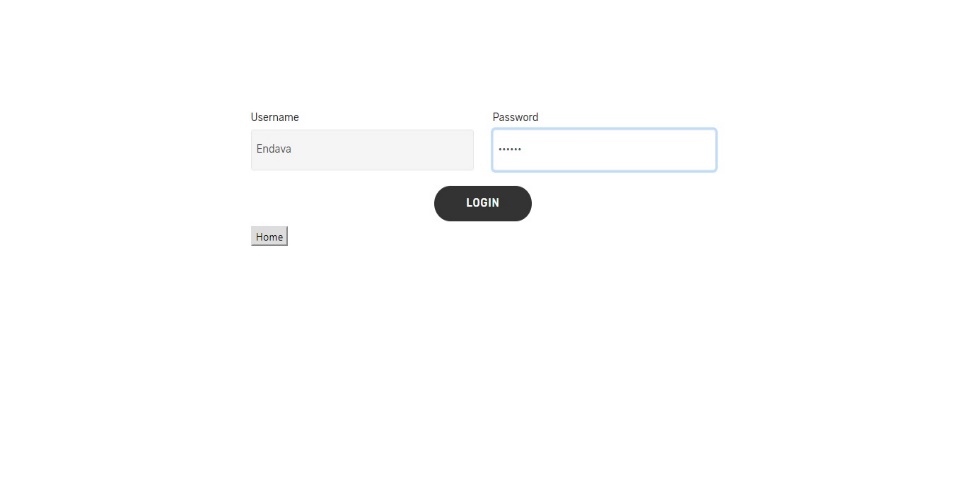
# Data Model

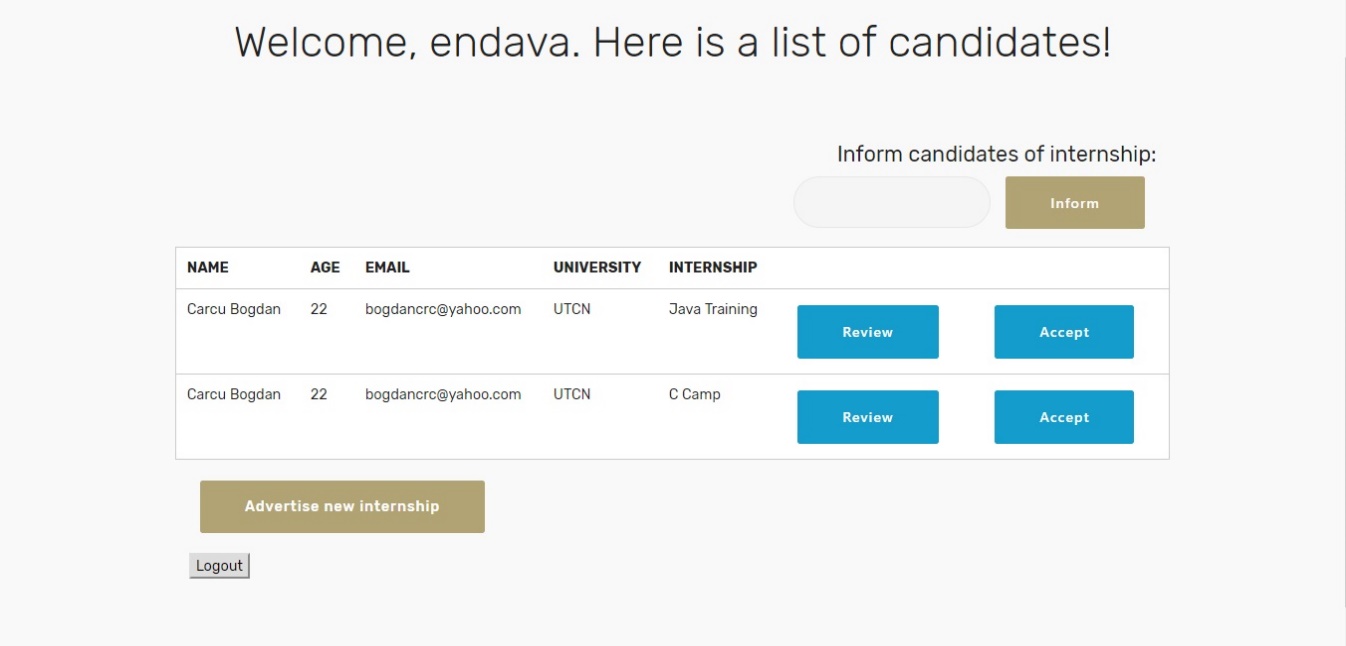


# Unit Testing

-Login as company user



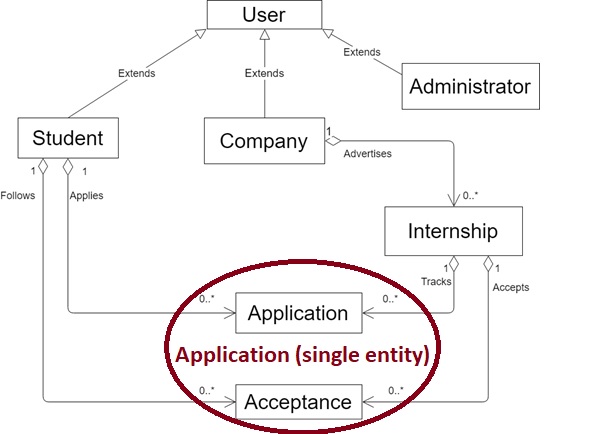




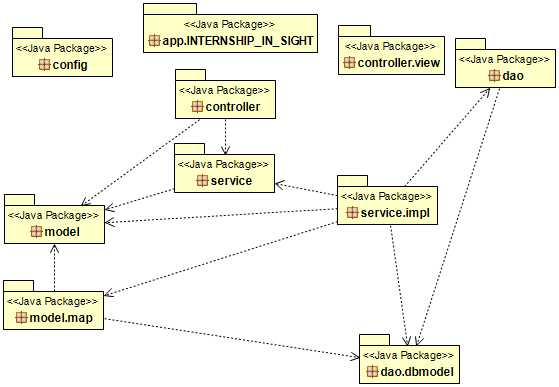
# Elaboration – Iteration 2

# Architectural Design Refinement

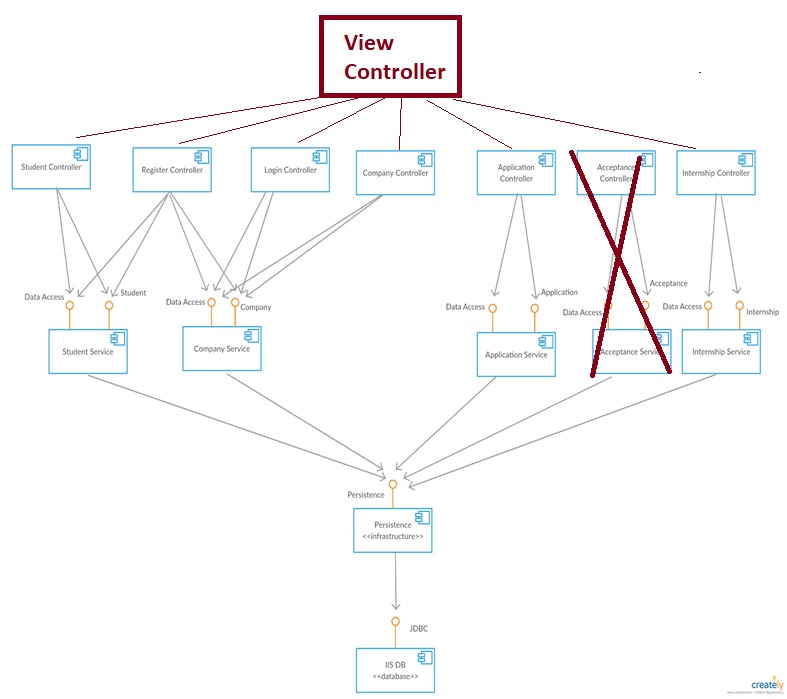
-Concept



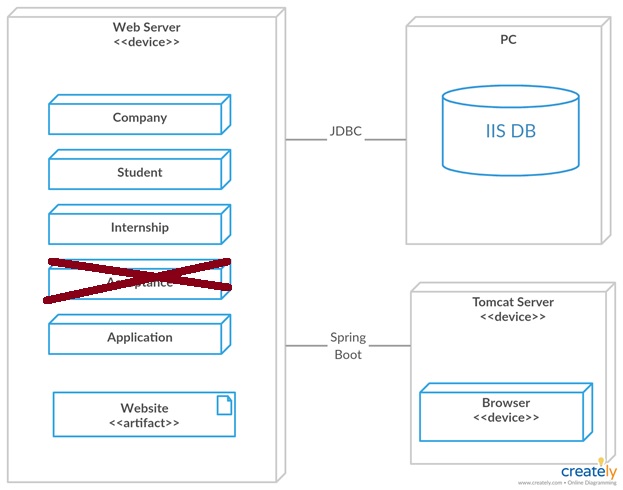
-Package



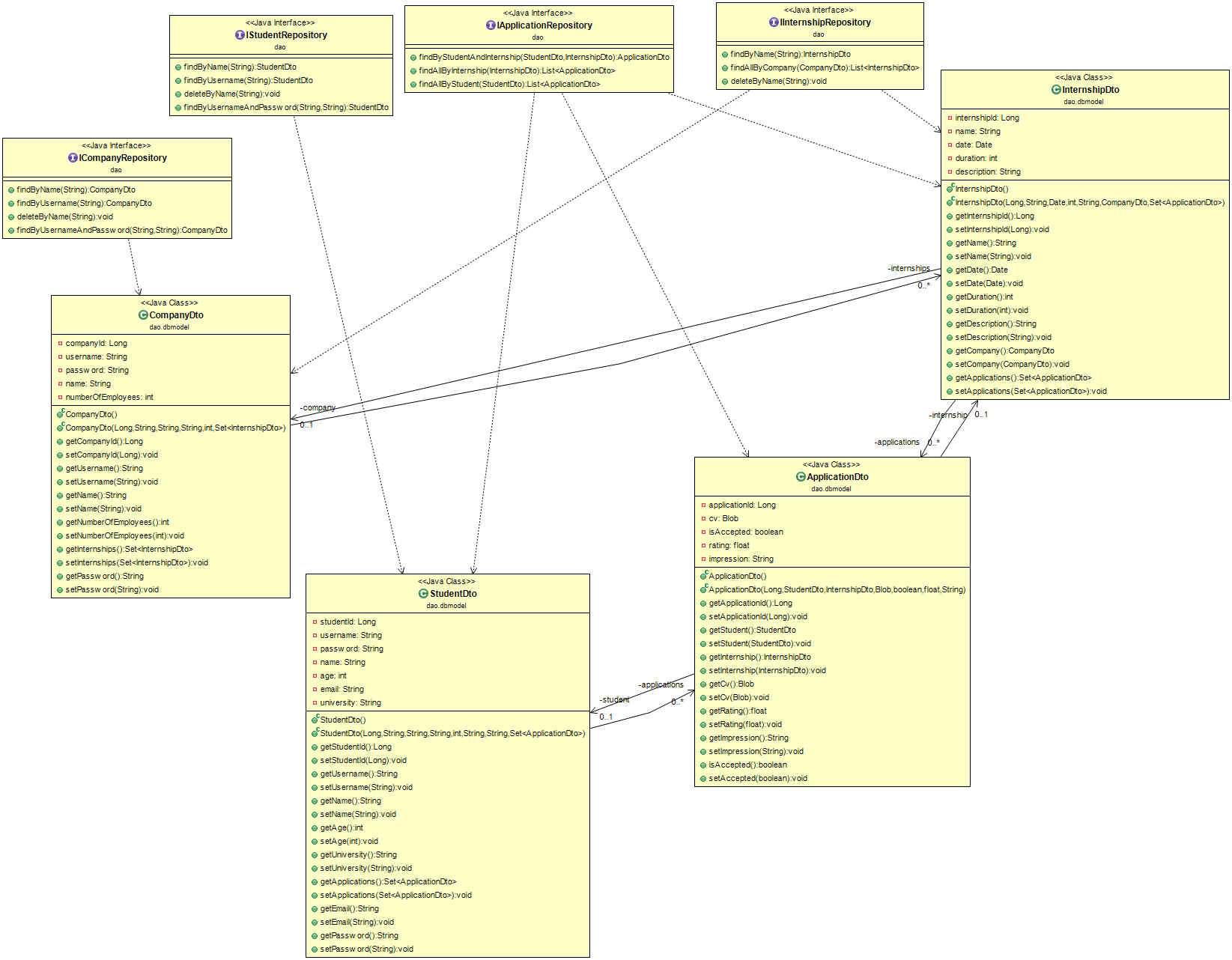
-Component

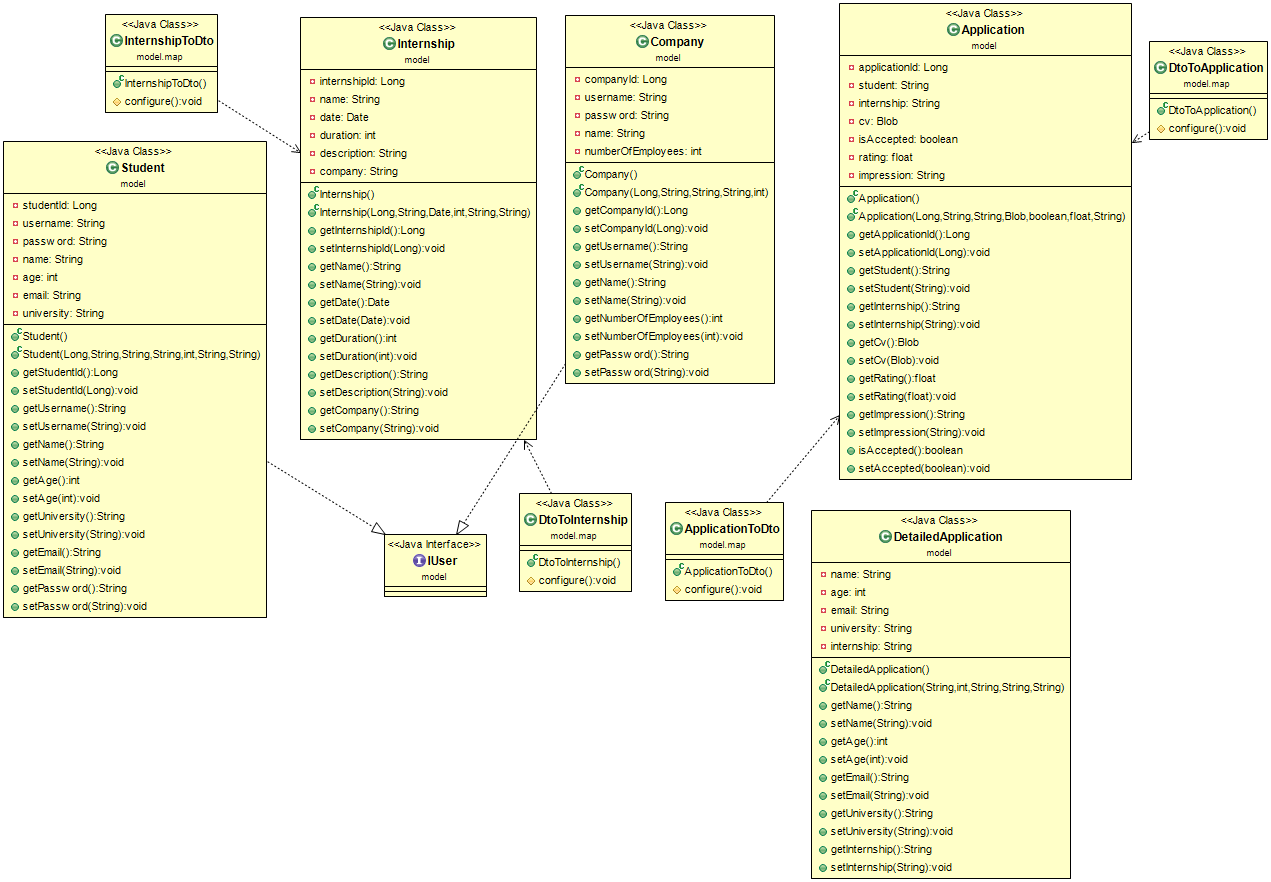


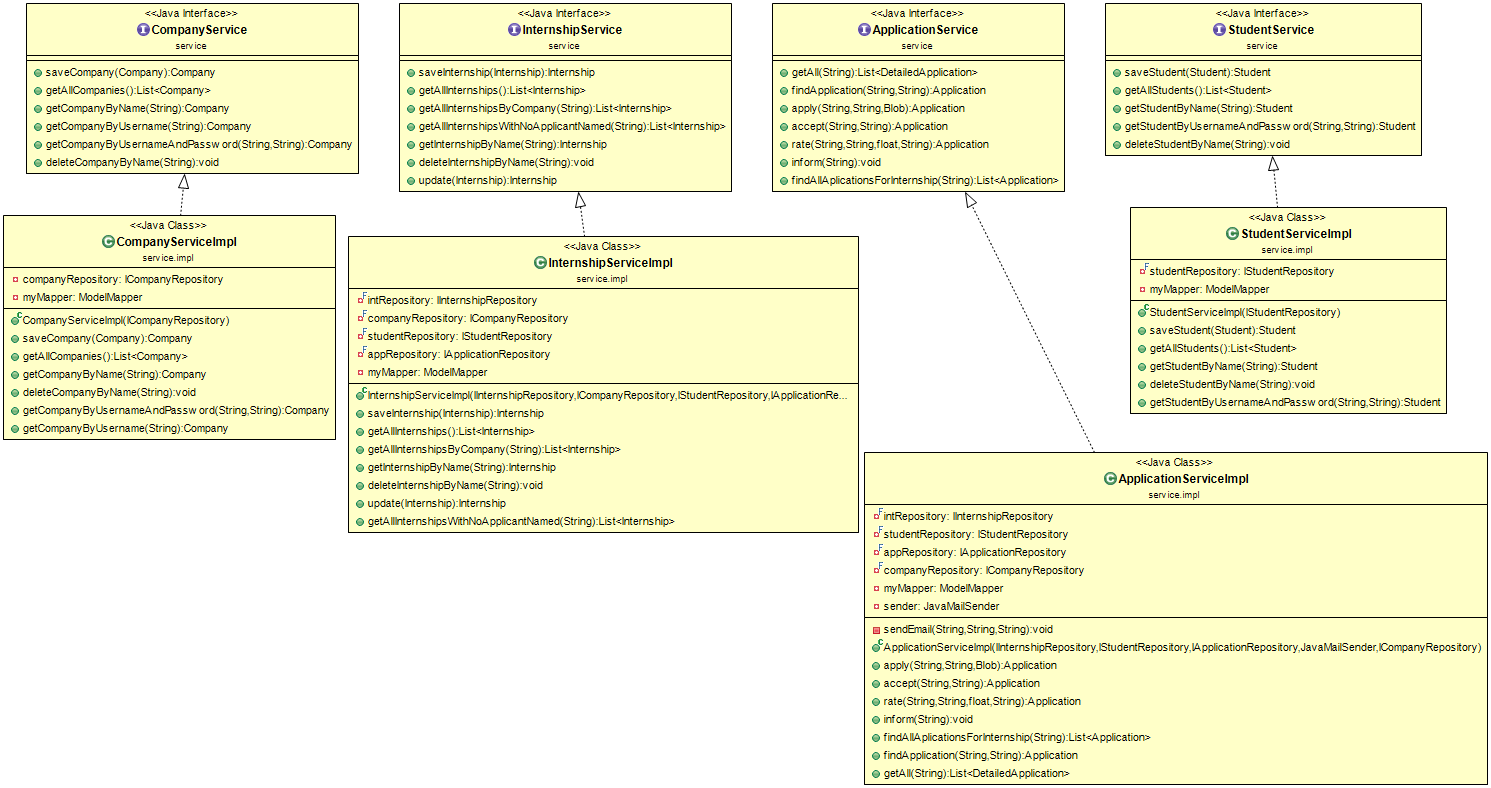
-Deployment

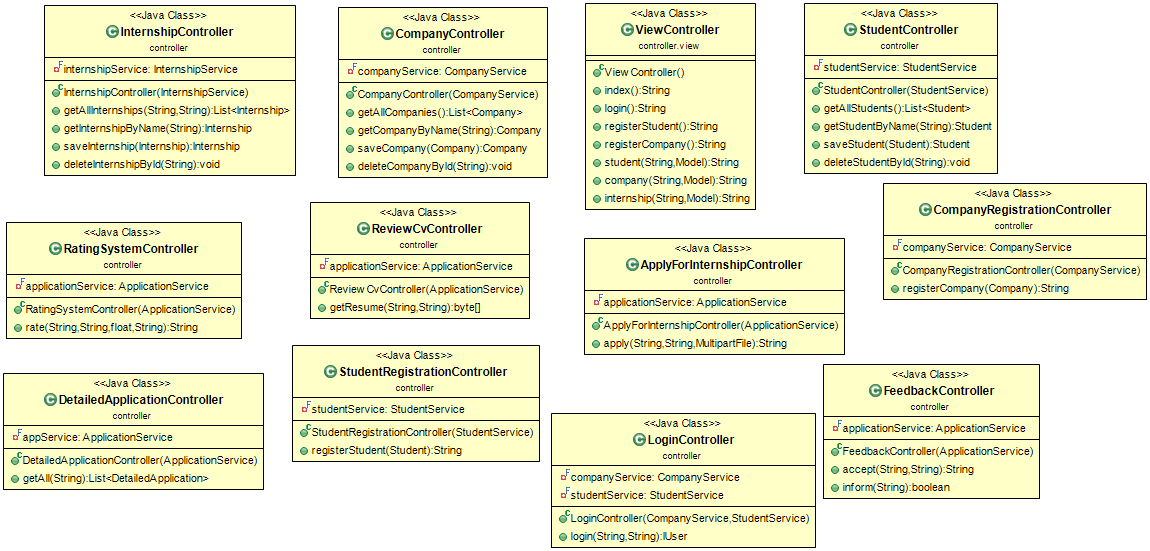


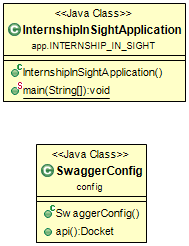
# Design Model Refinement

****

**

**

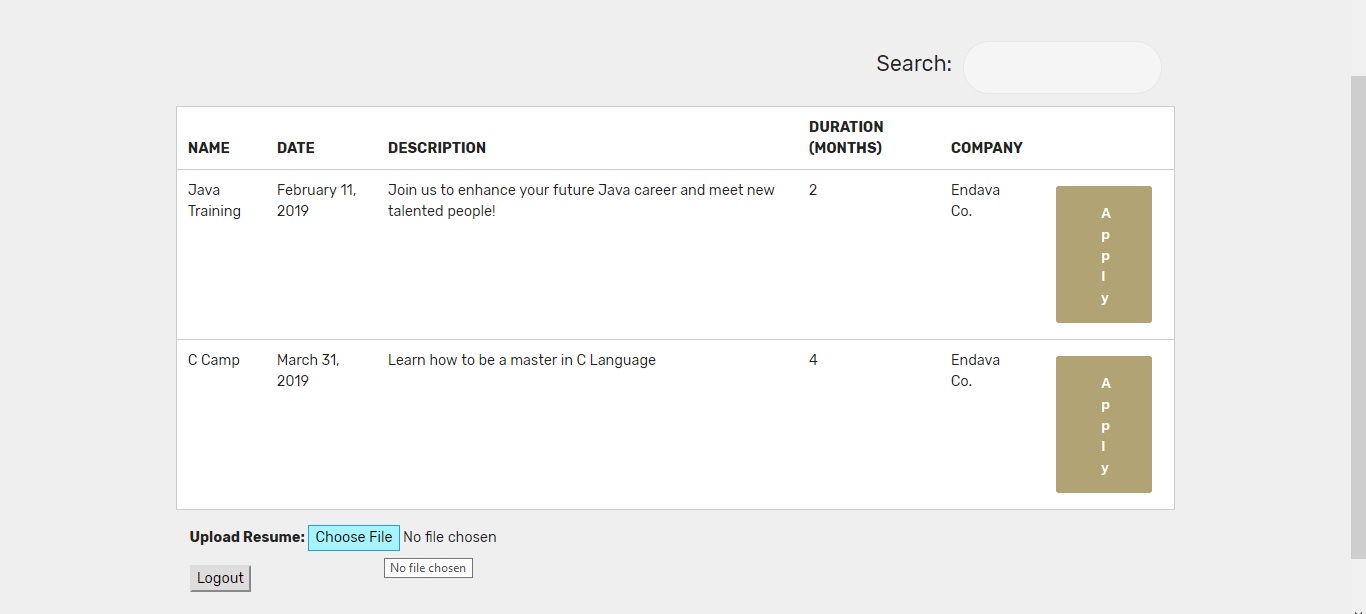
**

**

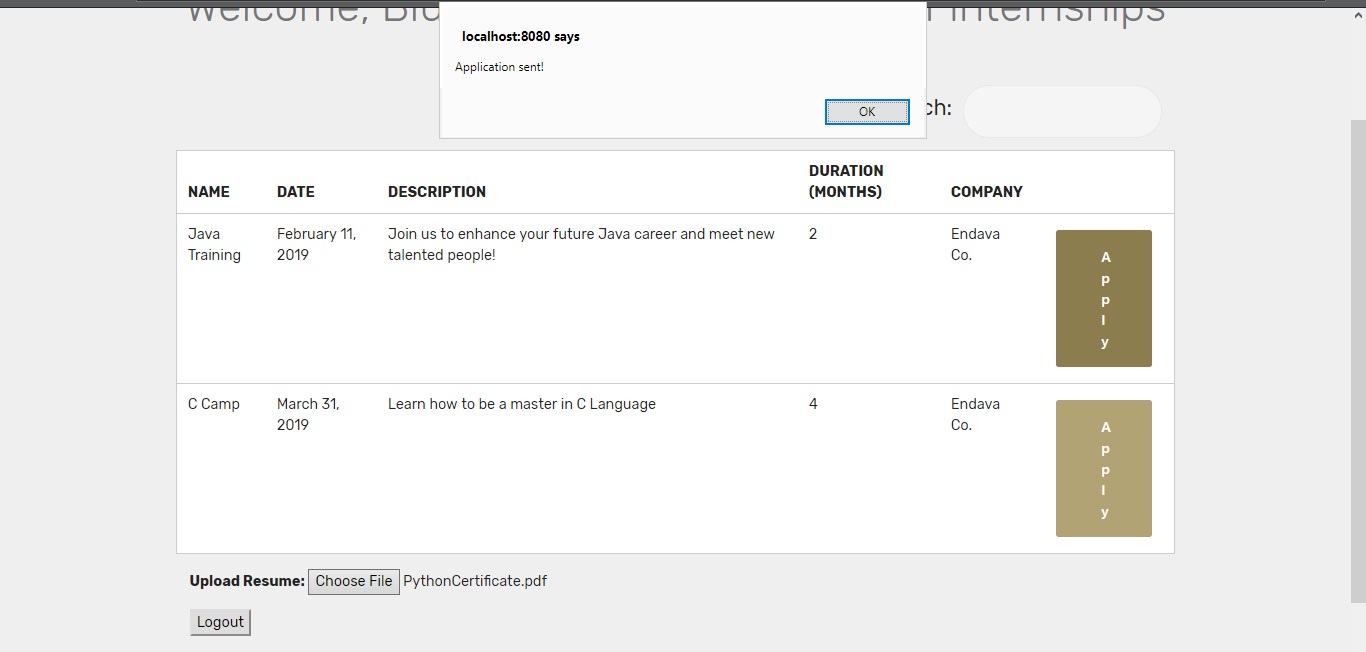
# Construction and Transition

# System Testing

-Upload resume



-Apply to wanted internship



# Future improvements

* Rating System
* Internship tables with better CRUD operations
* Search internships by company (add more filters)
* Security

# Bibliography

* Spring Boot 101 – Thymeleaf and AngularJS

<https://www.youtube.com/watch?v=SWcWY_CvoiM&list=PLVApX3evDwJ1i1KQYCcyS9hpSy_zOgU0Y>

<https://github.com/dangeabunea/RomanianCoderExamples/tree/master/BookingDemo>

* **Mobirise** (Free website maker): creating both desktop and mobile-friendly templates.

<https://mobirise.com>

* Diagrams

<https://creately.com>

<https://www.draw.io>

* MVC

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

* Setting up Swagger2 with a Spring REST API

<http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>