

|  |
| --- |
|  |

|  |
| --- |
| **FPT UNIVERSITY** |
| **SOFTWARE DESIGN**  **DESCRIPTION** |
| Internet-connected Devices Checking System |
|  |
| |  |  |  | | --- | --- | --- | | **IDCS TEAM** | | | | **Group Members** | Nguyễn Quý Đôn | SE04468 | | Nguyễn Quý Tuấn | SE04330 | | Phạm Công Minh | SE04098 | | **Supervisor** | Lecture : Phan Duy Hùng | | |
|  |

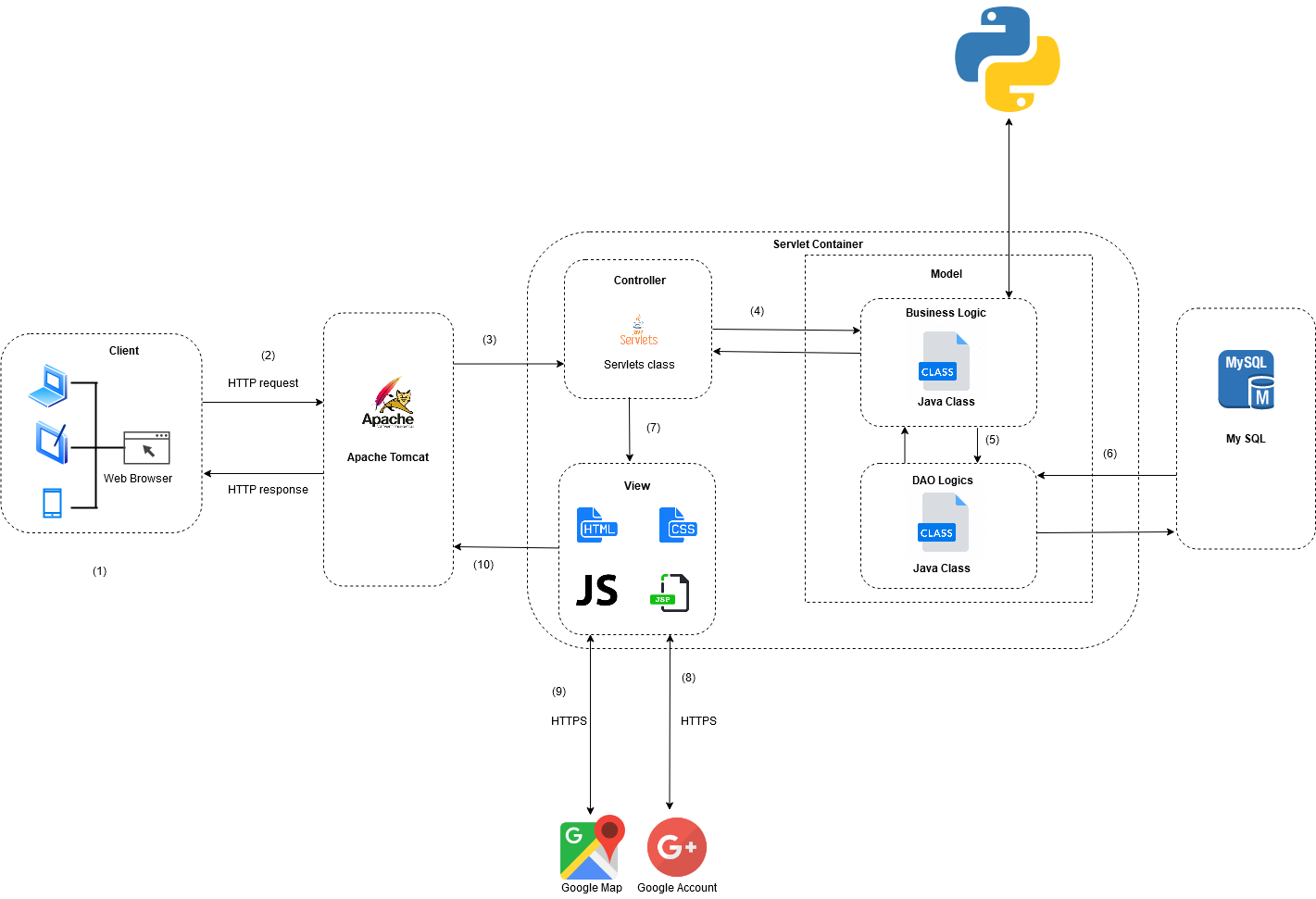
|  |
| --- |
| - Hanoi, 11/2019 - |

1. ***Purpose***

This chapter is to give developer team the overview and detailed design of what the system’s architecture is, and how they should be implemented. This chapter included are:

* Architecture overview
* Component diagram
* Detailed design
* Detailed description of components
* Database design

1. ***Architecture Overview***
   1. ***System Architecture***
   2. ***Diagram***



* 1. ***System Architecture Explanation***
     1. ***Apache Tomcat***



**The Apache Tomcat**® software is an open source implementation of the Java Servlet, Java Server Pages, Java Expression Language and Java Web Socket technologies. Tomcat provides a "pure Java" HTTP web server environment in which Java code can run.

* + 1. ***MySQL***

******

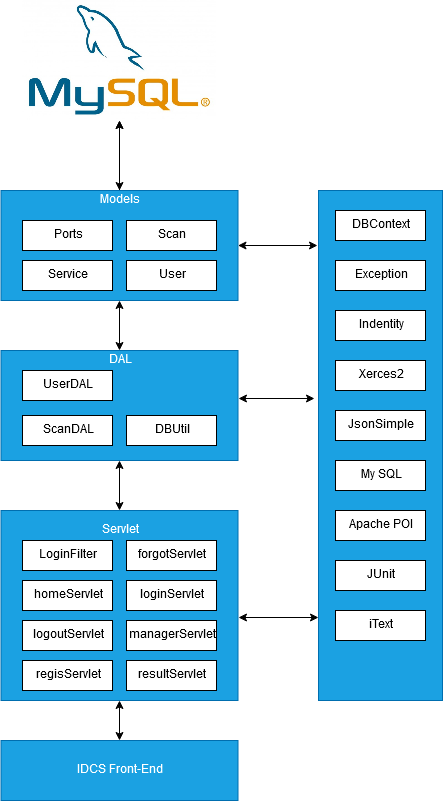
**MySQL** is the world's most popular open source database. With its proven performance, reliability and ease-of-use, **we use it to stores database.**

* + 1. ***Python***



**Python** is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. With Python, **we use it to scanning server and client target**

1. ***System Design***
   1. ***Architecture Layers Design***
      1. ***IDCS – Backend Layer Design***



* + 1. ***Explanation choice of architecture design***

For the back-end of IDCS, we have decided to use a 3-tier architecture style.

1. **Presentation tier**: The presentation tier is the front end layer in the 3-tier system and consists of the user interface. The main function of the interface is to translate tasks and result to something the user can understand.
2. **Logic tier:** This tier contains the functional business logic. Its consists of all the data of the application. Its written in Java and Python. It’s also moves and processes data between Data and Presentation layers.
3. **Data tier:** This tier comprises of the database/data storage system and data access layer. We use My SQL to store data. The information is then passed back to the logic tier for processing, and then back to the user.

This architecture makes system Scalability, Centralized Security and Fault Tolerance.

* + 1. ***Model-View-Controller Architecture***
       1. ***What is Model-View-Controller***

**Model-View-Controller** (usually knows as MVC) is a software design pattern that separates the business logic, presentation logic and data.

1. **Controller** acts as an interface between View and Model. Controller intercepts all the incoming request
2. **Model** is layer contain object information (Data), interacting with Database. Mainly responsible for object modeling.
3. **View** is interface of the system to interact directly with users.

MVC is a systematic way to use the application where the flow starts from the view layer, where the request is raised and processed in controller layer and sent to model layer to insert data and get back the success or failure message.

**Pros:**

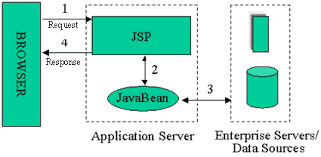
* Demonstrate professionalism in programming and object analysis, because it is divided into separate components, so independent activities help develop application faster, simpler and easier to upgrade and maintain.

**Cons:**

* As a very methodical working model, for small applications, using MVC is very time-consuming and causes a lot of complexity.
  + - 1. ***Two Types of MVC***

MVC is divided into two type: MVC1 and MVC2:

* + - * 1. **MVC Model 1:**

****

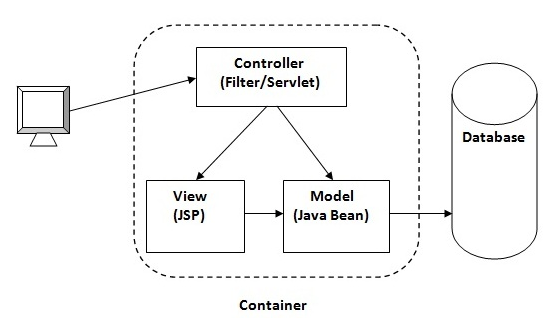
* In MVC model 1, Servlet technology does not create process, instead it creates a Thread to process the request. The advantage of creating a thread is that it does not allocate memory areas separately. Therefore, multiple consecutive request can easily be handle by the Servlet. The main problem with Servlet is that it needs to be recompiled if any code is modified.
* JSP handles most of the Servlet problems. You do need to re-deploy the application if the JSP page is modified. JSP provides support for developing web application using JavaBean, Custom Tag and JSLT custom tag so you can set specific logic with JSP, so testing and debugging become easier.

**Pros:**

* Easy and fast to develop web application

**Cons:**

* The navigation controls are not centralized as each page has its own logic to determine the next page. If the JSP page name is changed and is referenced by another page, then we must also change that name in the reference page.
* Time consuming. You need to spend more time developing Custom Tag custom tags in JSP.
  + - * 1. **MVC model 2**

****

* **Model:** State representation (data), and is the lowest layer in the sample. It responds to requests from the view and it also responds to directives from the controller to update itself.
* **View:** Responsible for displaying data in a specific format.
* **Controller:** Act as an Interface between View and Model. It intercepts and is responsible for responding to all requests (receiving inputs and commands) to the Model / View to perform the corresponding interactions. The controller receives input, it performs validation and then performs operations that modify the state of the Model.

With IDCS, we have decided to use MVC model 2.

* + - 1. ***Why MVC model 2?***
* **Reliability:** dividing each part separately helps us to modify each component separately, no effect, can be replaced into equivalent parts, can divide jobs in groups, compile independently, enhance integration with high correctness.
* **Compatibility:** able to use different types of technologies that are not dependent on us because we have been separated and conceptual for each type of component separately.
* **Reusable**: we can use components to re-divide in other applications or reuse multiple times in the same application, increasing efficiency in programming.
* Quick deployment and quick maintenance (quick deploy and easy maintenance): because components are independent of each other.