**TABLE OF CONTENTS**

**1** **INTRODUCTION 5**

1. Purpose 5

2. Scope 5

3. Definitions, Acronyms and Abbreviations 5

4. References 6

5. Overview 6

**2** **ARCHITECTURAL REPRESENTATION 7**

1.2.1 Presentation Layer 8

1.2.2 Business Layer 8

1.2.3 Data Access Layer 8

1.2.4 Data Layer 9

**3** **LOGICAL VIEW 9**

1. Overview 9

2. Architecturally Significant Design Packages/Components 9

2.3.1 JAVA01 presentation (JSPs, Actions) 9

2.3.2 Business Object 10

2.3.3 Value Objects 10

2.3.4 DAO 10

2.3.5 Hibernate Beans 10

2.3.6 Util 11

**APPENDIX A:** **LIST OF DESIGN PATTERNS AND FRAMEWORKS USED 12**

APPENDIX A.1: Struts 2 framework 12

APPENDIX A.2: SPRING Framework 13

APPENDIX A.3: HIBERNATE 15

Hibernate Core features: 15

1. **INTRODUCTION**
   1. **Purpose**

Tài liệu này cung cấp một cái nhìn tổng quan toàn diện về kiến trúc của các mô-đun Libra, sử dụng một số quan điểm kiến trúc khác nhau để mô tả các khía cạnh khác nhau của các mô-đun. Với mục đích nắm bắt và truyền đạt các quyết định kiến trúc quan trọng đã được thực hiện về các mô-đun, tài liệu đóng vai trò là cầu nối giữa các yêu cầu phần mềm và thiết kế chi tiết của Libra; nó cũng sẽ giúp kiến trúc sư phần mềm đảm bảo rằng các mô-đun được xây dựng sẽ đáp ứng nhu cầu của người dùng về chức năng, về nền tảng và công nghệ.

* 1. **Scope**

Từ quan điểm cấp cao, tài liệu xác định kiến trúc phần mềm giải quyết các yêu cầu của Libra trong các lĩnh vực sau: chức năng, tính khả dụng, độ tin cậy, khả năng mở rộng, khả năng bảo trì và khả năng quản lý.

* 1. **Definitions, Acronyms and Abbreviations**

| **#** | **Item** | **Description** |
| --- | --- | --- |
| 1 | Thymeleaf | Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text. |
| 2 | Repository | Repository hỗ trợ các câu lệnh truy vấn cơ sở dữ liệu, giúp việc lập trình dễ dàng hơn. |
| 3 | Service | Thực hiện các nghiệp vụ và xử lý logic |
| 4 | Controller | Tầng giao tiếp với bên ngoài và handler các request từ bên ngoài tới hệ thống. |
| 5 | Entity | là các đối tượng thuộc business nhưng tương ứng với table trong DB, có thể map vào DB được |
| 5 | API | Giao diện lập trình ứng dụng |
| 5 | Spring boot | Spring Boot cung cấp một nền tảng tốt cho các nhà phát triển Java để phát triển một ứng dụng production-grade spring và cấp sản xuất mà bạn có thể chạy. Bạn có thể bắt đầu với các cấu hình tối thiểu mà không cần thiết lập toàn bộ cấu hình Spring |
| 6 | HIBERNATE | Hibernate là một dịch vụ truy vấn và tính bền bỉ đối tượng / quan hệ mạnh mẽ, hiệu suất cao. Hibernate cho phép các nhà phát triển phát triển các persistent class theo hướng đối tượng - bao gồm liên kết, kế thừa, đa hình, thành phần và tập hợp. Hibernate cho phép các nhà phát triển thể hiện các truy vấn HQL của riêng nó, cũng như trong native SQL. |
| 7 | Spring Data JPA | **Spring Data** là một module của **Spring Framework**. Mục đích của **Spring Data** **JPA**làgiảm thiểu việc thực hiện quá nhiều bước để có thể implement được JPA. Spring Data JPA là một phần của Spring Data và nó hỗ trợ Hibernate 5, OpenJPA 2.4 và EclipseLink 2.6.1. |

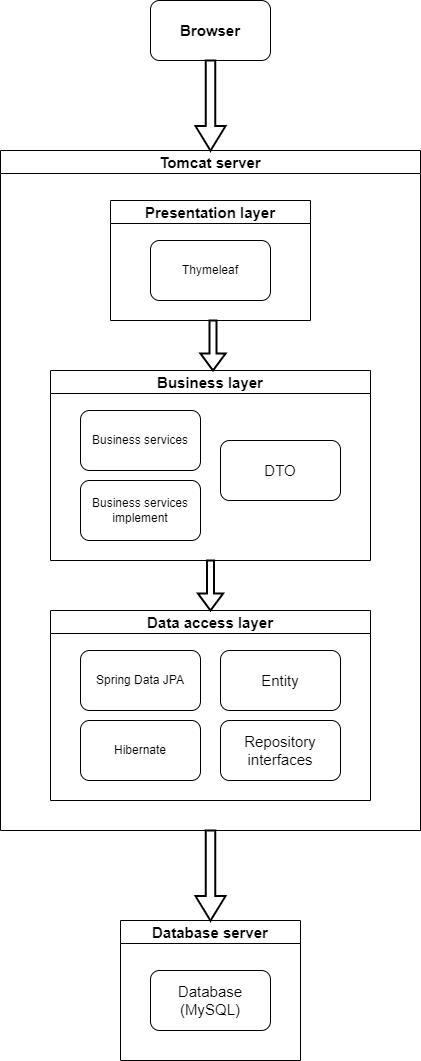
* 1. **References**

| **#** | **Name** | **Version** | **Date** | **Author** |
| --- | --- | --- | --- | --- |
| 1 | libra.docx | 1.0 | 05/05/2022 | CUSTOMER |

* 1. **Overview**

Các phần sau sẽ cung cấp cái nhìn sâu sắc hơn về kiến trúc mô-đun Libra, giải thích chức năng mà kiến trúc có thể cung cấp và cũng chỉ ra khả năng mở rộng của kiến trúc để phù hợp với những thay đổi sau này từ yêu cầu của người dùng hoặc những thay đổi trong giao diện hệ thống với bên thứ ba khác ứng dụng / thành phần. Nhìn kiến trúc từ các góc độ chính như Chức năng và Logic, chúng ta có thể đảm bảo kiến trúc sẽ thỏa mãn tất cả các yêu cầu đã xác định và vẫn cho phép mở rộng cần thiết sau này. Tất cả các công nghệ liên quan sẽ được áp dụng cho một mục đích cụ thể của kiến trúc phần mềm cũng sẽ được trình bày ở dưới đây.

1. **ARCHITECTURAL REPRESENTATION**



Hình 1: Kiến trúc hệ thống

### Presentation Layer

Lớp này kiểm soát việc hiển thị cho người dùng cuối. Đối với presentation layer của Libra, Spring Boot framework được sử dụng. (Kiến trúc Spring Boot được mô tả chi tiết hơn trong APPENDIX A.1:).

Khung chịu trách nhiệm cho:

* Quản lý các request / response tới client.
* Kiểm soát hiển thị cho người dùng cuối.
* Lắp ráp một model có thể được trình bày trong một view.
* Thực hiện xác thực giao diện người dùng.
* Cung cấp controller để ủy quyền các cuộc gọi đến business logic và các quy trình ngược dòng khác.
* Xử lý các ngoại lệ từ các lớp khác.

### Business Layer

Layer này quản lý các quy tắc xử lý business và logic.

* Xử lý business logic ứng dụng và business validation.
* Quản lý các transaction.
* Cho phép interface tương tác với các layer khác.
* Quản lý dependencies giữa các đối tượng business level.
* Thêm tính linh hoạt giữa presentation và persistence layer để chúng không giao tiếp trực tiếp với nhau.
* Hiển thị ngữ cảnh cho business layer từ presentation layer để nhận business services.
* Quản lý việc triển khai từ business logic đến persistence layer.

### Data Access Layer

Layer này quản lý quyền truy cập vào bộ nhớ persistent. Lý do chính để tách quyền truy cập dữ liệu khỏi phần còn lại của ứng dụng là việc chuyển đổi nguồn dữ liệu và chia sẻ Data Access Objects (DAO) giữa các ứng dụng dễ dàng hơn.

* Layer này quản lý việc đọc, ghi, cập nhật và xóa dữ liệu được lưu trữ.
* HIBERNATE (như được mô tả chi tiết trong APPENDIX A.3) được sử dụng như một persistence framework quản lý ánh xạ O-R một cách liền mạch và cho phép các nhà phát triển nhanh chóng xây dựng các ứng dụng kết hợp các khía cạnh tốt nhất của công nghệ đối tượng và cơ sở dữ liệu quan hệ.

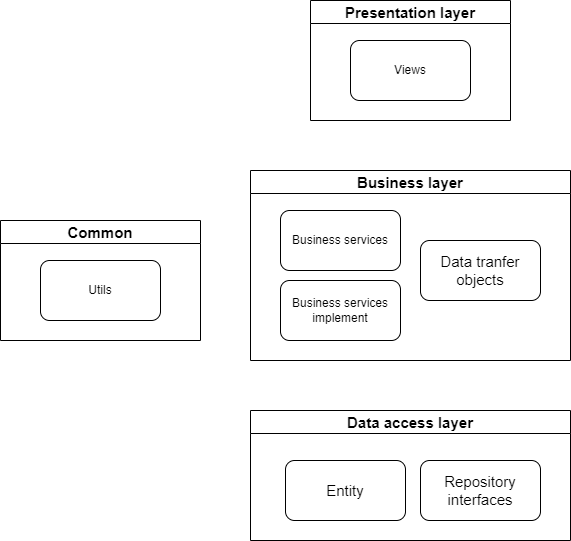
### Data Layer

Trong Libra, bộ nhớ được quản lý bởi cơ sở dữ liệu quan hệ. Cơ sở dữ liệu My SQL được sử dụng cho lớp này để cung cấp việc quản lý dữ liệu được lưu trữ.

1. **LOGICAL VIEW** 
   1. **Overview**

Phần này mô tả các package / component quan trọng của kiến trúc ở module Libra với thiết kế kiến trúc cũng như trách nhiệm của chúng.

* 1. **Architecturally Significant Design Packages/Components**



Hình 2 – Kiến trúc packages

### Package presentation layer(Views)

Package name: com.libra.webapp.view

Package này bao gồm việc triển khai cho Presentation Layer để xử lý hiển thị cho người dùng cuối.

### Package Business Layer (Services)

Package name: com.nlu.service

Package này bao gồm việc thực hiện các business object. **Business Object** (BO) layer được sử dụng để thực hiện các hoạt động business. Business Object layer sẽ truy cập DAO bằng cách triển khai Dependency Injection pattern. Các Transaction phải được quản lý trong business layer này.

* **Spring Transaction demarcation:** Sử dụng quản lý transaction khai báo Spring lightweight nơi có thể khai báo cài đặt transaction trên toàn hệ thống. Tức là đối với tất cả các phương thức đối tượng business bắt đầu bằng 'insert' sẽ bắt đầu transaction nhưng đối với tất cả các phương thức đối tượng business bắt đầu bằng "find" sẽ không bắt đầu transaction. Điều này rất linh hoạt và sẽ được áp dụng bằng Hibernate.

### Package Data access Layer (Repositories)

Package name: com.nlu.repository

Package này bao gồm việc triển khai DAO Java pattern. Sử dụng mẫu thiết kế DAO ở đây để làm cho ứng dụng linh hoạt hơn để truy cập cơ sở dữ liệu. DAO bao gồm các chức năng cơ bản để làm việc với cơ sở dữ liệu: chọn, chèn, cập nhật, xóa

### Util

Package name: com.libra.util

Package này bao gồm tất cả các util class Java sẽ được sử dụng trong các mô-đun.

1. **List of Design Patterns and Frameworks Used**

Phần này mô tả chi tiết các mẫu thiết kế và framework được sử dụng trong kiến trúc ứng dụng được đề xuất cho DWA.

* 1. ***Thymeleaf***

*Thymeleaf là template engine Java phía server-side hiện đại cho cả môi trường web và môi trường độc lập.*

*Mục tiêu chính của Thymeleaf là mang lại các mẫu tự nhiên thanh lịch cho quy trình phát triển của bạn - HTML có thể được hiển thị chính xác trong các trình duyệt và cũng hoạt động như các static prototype, cho phép cộng tác mạnh mẽ hơn trong các nhóm phát triển.*

*Với các mô-đun dành cho Spring Framework, một loạt các tích hợp với các công cụ yêu thích của bạn và khả năng cắm vào chức năng của riêng bạn, Thymeleaf lý tưởng cho việc phát triển web HTML5 JVM hiện đại - mặc dù còn nhiều điều hơn thế nữa*

* 1. ***Spring boot Framework***

**Inversion of control and dependency injection**

***Inversion of Control (IoC)*** *còn được gọi là* ***Dependency injection (DI)****. Spring container sử dụng* ***Dependency Injection (DI)*** *để quản lý các thành phần tạo nên một ứng dụng và các đối tượng này được gọi là* ***Spring Beans****. Spring triển khai DI bằng tệp cấu hình XML hoặc các chú thích. Container chịu trách nhiệm quản lý vòng đời của đối tượng (tạo các đối tượng, kết nối chúng lại với nhau, cấu hình chúng và quản lý vòng đời hoàn chỉnh của chúng từ khi tạo cho đến khi bị phá hủy). Mục tiêu chính của* ***Inversion of control*** *và* ***Dependency Injection*** *là loại bỏ các phần phụ thuộc của một ứng dụng. Điều này làm cho hệ thống được tách rời và bảo trì tốt hơn*.

**Spring boot framework**

Là một dự án của nền tảng Spring hoặc hệ sinh thái Spring. Đó là một cách nhằm mục đích giúp các nhà phát triển bắt đầu với Spring framework với nỗ lực tối thiểu, dễ dàng hơn và thuận tiện hơn - vì việc phát triển với Spring framework thuần túy được biết đến như là boilerplate với rất nhiều cấu hình XML và mớ hỗn độn dependency.

Là một công cụ giúp phát triển ứng dụng web và microservices bằng Spring Framework nhanh hơn và dễ dàng hơn thông qua ba khả năng cốt lõi:

* Tự động cấu hình
* Một cách tiếp cận kiên quyết đối với cấu hình
* Khả năng tạo các ứng dụng độc lập

Các tính năng này phối hợp với nhau để cung cấp cho bạn một công cụ cho phép bạn thiết lập một ứng dụng dựa trên Spring với cấu hình và thiết lập tối thiểu.

**Spring Boot Advantages**

1. Tạo các ứng dụng Spring độc lập
2. Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần triển khai tệp WAR)
3. Cung cấp các POM “starter”' được xác định để đơn giản hóa cấu hình Maven của bạn
4. Tự động cấu hình Spring bất cứ khi nào có thể
5. Cung cấp các tính năng sẵn sàng cho sản xuất như chỉ số, kiểm tra tình trạng và cấu hình bên ngoài
6. Hoàn toàn không tạo mã và không yêu cầu cấu hình XML
   1. ***HIBERNATE***

Hibernate nổi tiếng và được hỗ trợ bởi SPRING framework, với các tính năng chính sau:

Hibernate là một đối tượng mạnh mẽ, hiệu suất cao / dịch vụ truy vấn và tính bền vững quan hệ. Hibernate cho phép các nhà phát triển phát triển các lớp liên tục theo thành ngữ hướng đối tượng - bao gồm liên kết, kế thừa, đa hình, thành phần và tập hợp. Hibernate cho phép các nhà phát triển thể hiện các truy vấn trong tiện ích mở rộng SQL di động (HQL) của riêng nó, cũng như trong SQL gốc hoặc với API mẫu và tiêu chí hướng đối tượng

**Hibernate Core features:**

* **Mô hình lập trình tự nhiên** - Hibernate hỗ trợ hướng đối tượng tự nhiên; kế thừa, đa hình, thành phần và khung bộ sưu tập Java
* **Hỗ trợ cho các mô hình đối tượng-grained** - nhiều loại ánh xạ phong phú cho các bộ sưu tập và các đối tượng phụ thuộc
* **No build-time bytecode enhancement** - không có bước tạo mã bổ sung hoặc xử lý mã byte trong quy trình xây dựng của bạn
* **Khả năng mở rộng cực cao** - Hibernate cực kỳ hiệu quả, có kiến trúc bộ đệm hai lớp và có thể được sử dụng trong một cụm
* **Các tùy chọn truy vấn** - Hibernate giải quyết cả hai mặt của vấn đề; không chỉ cách đưa các đối tượng vào cơ sở dữ liệu mà còn cách lấy lại chúng
* **Hỗ trợ cho "conversations"** - Hibernate hỗ trợ cả bối cảnh tồn tại lâu dài, tách / gắn lại các đối tượng và tự động xử lý khóa lạc quan
* **Free/open source** - Hibernate được cấp phép theo LGPL (Giấy phép Công cộng GNU ít hơn)