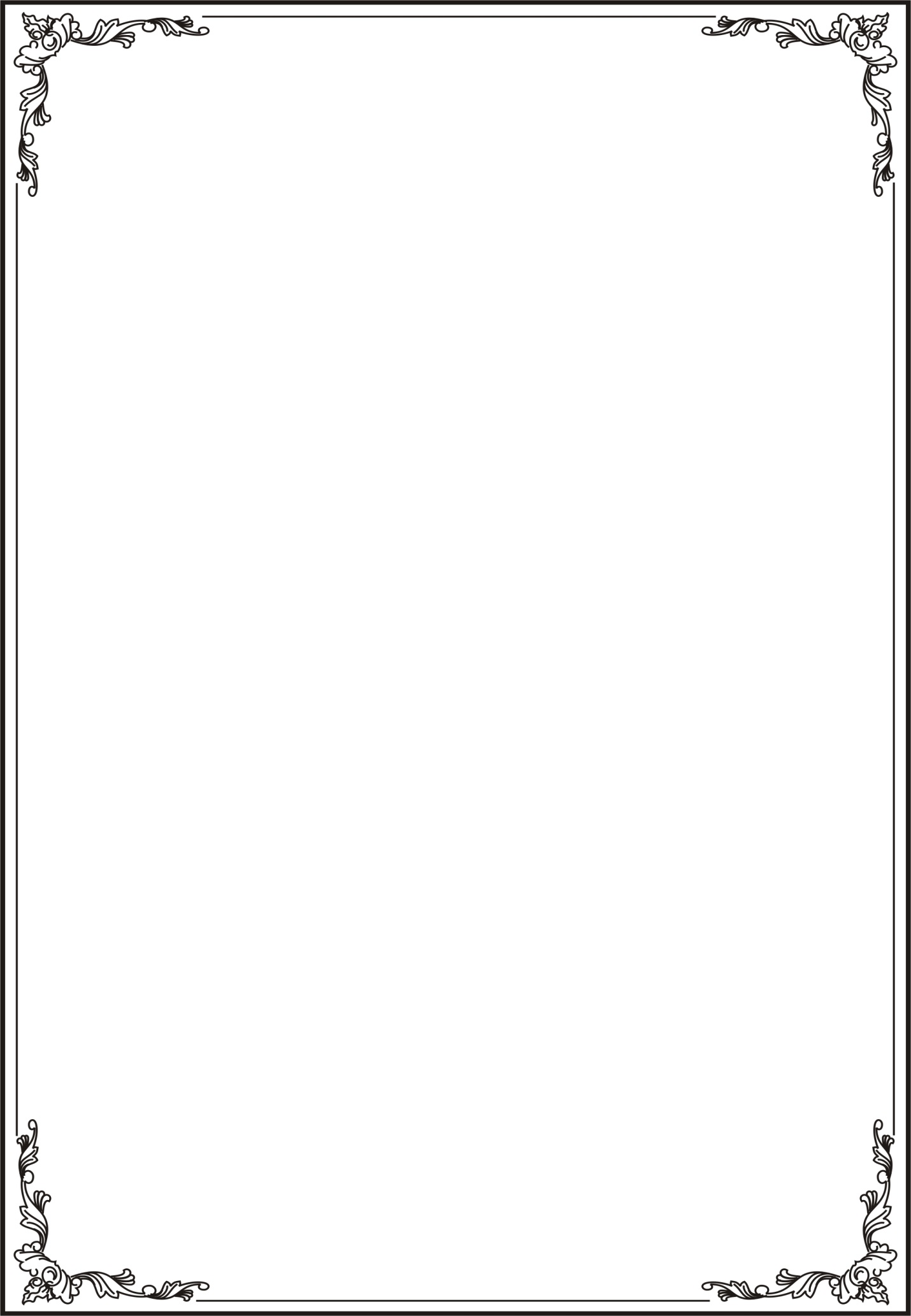
****

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**---------------🙦 🕮 🙤---------------

*TP. Hồ Chí Minh, tháng 1, năm 2021*

***Đề tài:***

**WEBSITE TỪ ĐIỂN**

**BÁO CÁO ĐỒ ÁN CNTT**

***Giảng viên hướng dẫn:* Trần Công Tú**

***Lớp:* 18110CL1C**

***Sinh viên:***

**Nguyễn Lâm Gia Khang 18110132**

**Nguyễn Dương Đạt 18110092**

**Mục lục**

[Mục lục hình ảnh 2](#_Toc60601051)

[NỘI DUNG 3](#_Toc60601052)

[I. Giới thiệu đề tài 3](#_Toc60601053)

[1. Lý do chọn đề tài 3](#_Toc60601054)

[2. Mục tiêu đề tài 3](#_Toc60601055)

[3. Phương pháp thực hiện 3](#_Toc60601056)

[II. Quá trình thực hiện 4](#_Toc60601057)

[1. Thiết kế giao diện 4](#_Toc60601058)

[1.1 Trang tìm kiếm 4](#_Toc60601059)

[1.2. Trang quản lí các từ 6](#_Toc60601060)

[1.3. Trang thêm một từ mới: 8](#_Toc60601061)

[2. Nội dung chính 9](#_Toc60601062)

[2.1. Xây dựng mô hình MVC và 3-tier 9](#_Toc60601063)

[2.2. Ngôn ngữ, hệ quản trị cơ sở dữ liệu và công nghệ được sử dụng: 10](#_Toc60601064)

[2.3. Cấu trúc dữ liệu và giải thuật 10](#_Toc60601065)

[2.4. Ứng dụng giải quyết bài toán từ điển 10](#_Toc60601066)

[2.5. Thiết kế lớp 13](#_Toc60601067)

[2.6. Chi tiết các lớp 14](#_Toc60601068)

[3. Cài đặt và kiểm thử 28](#_Toc60601069)

[3.1. Tìm kiếm từ 28](#_Toc60601070)

[3.2. Cập nhật từ 30](#_Toc60601071)

[3.3. Xoá từ 32](#_Toc60601072)

[III.Phân công công việc 33](#_Toc60601073)

[IV.Kết luận 33](#_Toc60601074)

[4.1. Đánh giá mức độ hoàn thành 33](#_Toc60601075)

[4.2. Khó khăn 34](#_Toc60601076)

[4.3. Hướng phát triển 34](#_Toc60601077)

[V. Phụ lục 35](#_Toc60601078)

[VI. Tài liệu tham khảo 50](#_Toc60601079)

# Mục lục hình ảnh

[Hình 1.1: Trang tìm kiếm từ 4](#_Toc59479857)

[Hình 1.2: Thiết kế trang web chỉnh sửa 6](#_Toc59479858)

[Hình 1.3: Trang thêm một từ mới 8](#_Toc59479859)

Hình 2.1: Mô hình MVC và 3-tier…………………………………………………….. 9 Hình 2.2: Mô tả cơ sở dữ liệu…………………………………………………………11  
Hình 3.1.1: Gợi ý tra từ……………………………………………………………… 29  
Hình 3.1.2: Trang kết quả tra từ……………………………………………………... 29  
Hình 3.1.3: Trang không tìm thấy từ cần tra………………………………………….30  
Hình 3.2.1: Trang thêm từ mới………………………………………………………..30  
Hình 3.2.2: Tìm kiếm lại từ đã thêm………………………………………………….31  
Hình 3.3: Tìm kiếm lại từ đã xóa…………………………………………………......32

# NỘI DUNG

## I. Giới thiệu đề tài

1. Lý do chọn đề tài

Toàn cầu hóa đang là xu thế chung của thế giới ngày nay. Việt Nam hiện đang là một quốc gia đang phát triển, đang trong xu thế mở cửa, hội nhập với nền kinh tế toàn cầu hóa của thế giới. Một trong những công cụ quan trọng để cho một quốc gia có thể hội nhập vào nền kinh tế thế giới đó chính là trình độ ngoại ngữ của những người dân nước ấy. Đặc biệt là các thế hệ học sinh, sinh viên mầm non tương lai của đất nước việc học ngoại ngữ đối với họ là một điều rất quan trọng. Trong các ngôn ngữ chính trên thế giới hiện nay thì có thể nói tiếng Anh là phổ cập nhất.

Để học tốt tiếng Anh thì không gì bằng một quyển từ điển tốt. Xuất phát từ nhu cầu thực tiễn đó nhóm lập trình của nhóm thực hiện đã xây dựng một chương trình từ điển Anh-Việt và Việt-Anh.

Do thao tác chính trên phần mền từ điển là việc tra từ chính vì thế nhóm đã chọn cây nhị phân tìm kiếm làm cấu trúc dữ liệu chính cho đề tài này để cho việc tìm kiếm từ trở nên nhanh hơn và dễ dàng hơn.

1. Mục tiêu đề tài

Chương trình có một giao diện đơn giản, thân thiện cho phép người dùng dễ dàng sử dụng. Để cho việc tra tra cứu từ tiếng Anh một cách dễ dàng hơn. Ngoài việc tra cứu từ phần mền từ điển còn cho phép người dung cập nhật từ, bổ sung từ, thêm từ hoặc xóa từ tiếng Anh trong từ điển.

1. Phương pháp thực hiện

Sử dụng cây nhị phân, các hàm của cây nhị phân tìm kiếm để cài đặt chương trình tra từ điển kết hợp một số chức năng về quản lý từ điển. Bộ dữ liệu gồm các từ thông dụng trong tiếng Anh. Chương trình sử dụng giao diện web, từ cần tra được nhập từ bàn phím. Bên cạnh đó hiện thực hóa trên web để có thể mang đến sự tiện lợi trong quá trình sử dụng. Thiết kế và thực hiện theo mô hình MVC và 3-tier để có thể nâng cấp và sửa chữa trong quá trình sử dụng.

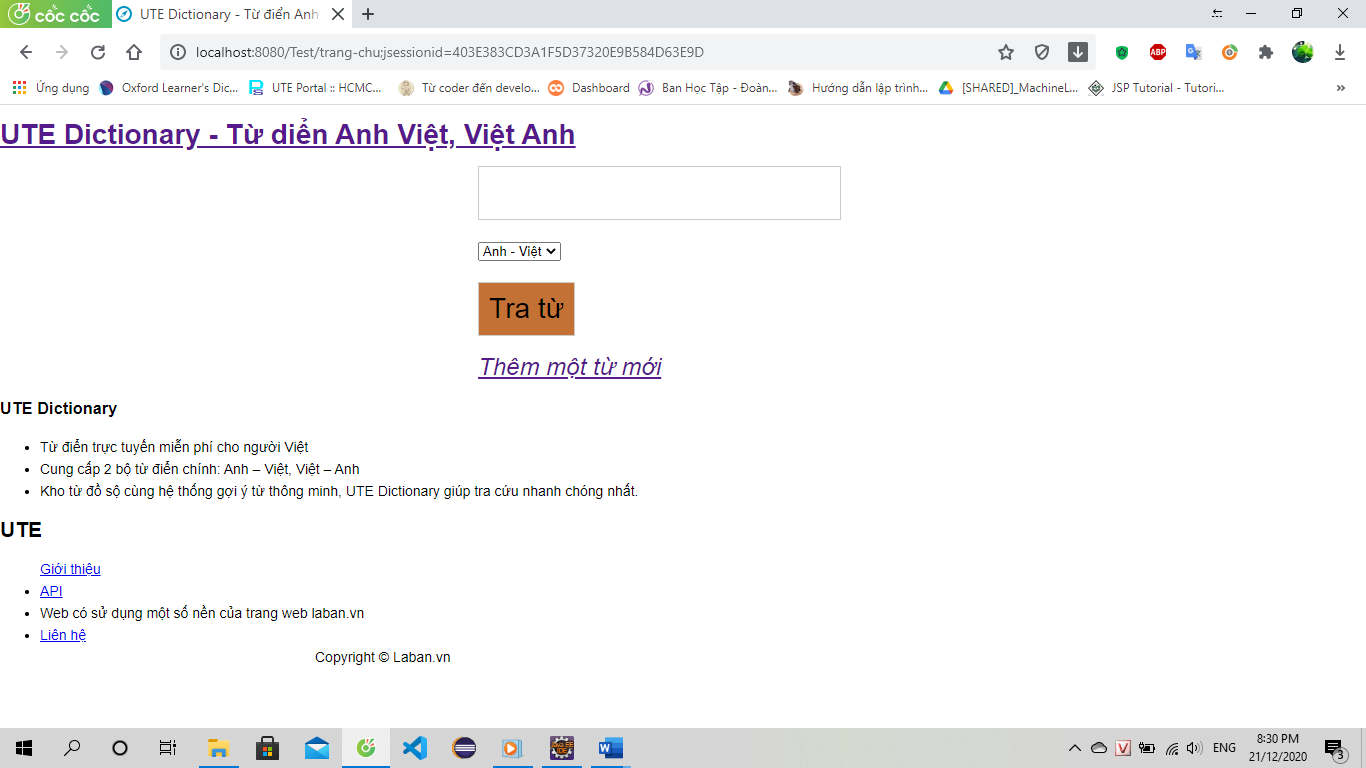
## II. Quá trình thực hiện

1. Thiết kế giao diện

Giao diện của từ điển UTE Dictionary được thiết kế đơn giản, thân thiện để người dùng dễ dàng sử dụng. Phần giao diện này được tổ chức thành 2 phần: từ điển Anh-Việt và Việt-Anh nhằm thực hiện các chức năng riêng: tra từ, tạo từ mới, cập nhật từ đã có và xóa từ trong file.

1.1 Trang tìm kiếm

Giao diện tìm kiếm là giao diện chính của từ điển dùng để nhập từ cần tra, tra từ và chuyển sang các giao diện chức năng khác của từ điển.



Hình 1.1: Trang tìm kiếm từ

Để làm được giao diện này, khi thiết kế home.jsp, home2.jsp (dùng home.jsp để tra từ Anh-Việt và page còn lại tra từ Việt-Anh) của chương trình, nhóm đã sử dụng jsp (bao gồm html và java), css, javascript, trang bao gồm các thành phần sau đây:

Thành phần phục vụ cho chức năng tra từ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jsp (HTML+ java)+ css+ javacript | Header | Thành phần | Tên | Mô tả |
| Class | logo | Class của css chứa file ảnh logo. |
| Thẻ a |  | Nằm trong logo chứa link dẫn về trang chủ. |
| Body | Textbox | Term | Phục vụ cho việc nhập từ từ bàn phím. |
| Select tag | searchOptions | Lựa chọn từ điển cần tra. |
| Submit |  | Trả về dữ liệu: textbox và select tag cho form. |
| Form |  | Đưa dữ liệu sang servlet để xử lý. |
| Autocomplete |  | Xử dụng thư viện của javacript là jquery để hiện gợi ý giúp người dùng dễ dàng tra được từ. |
| Footer |  |  | Chứa các thông tin giới thiệu, đường link. |

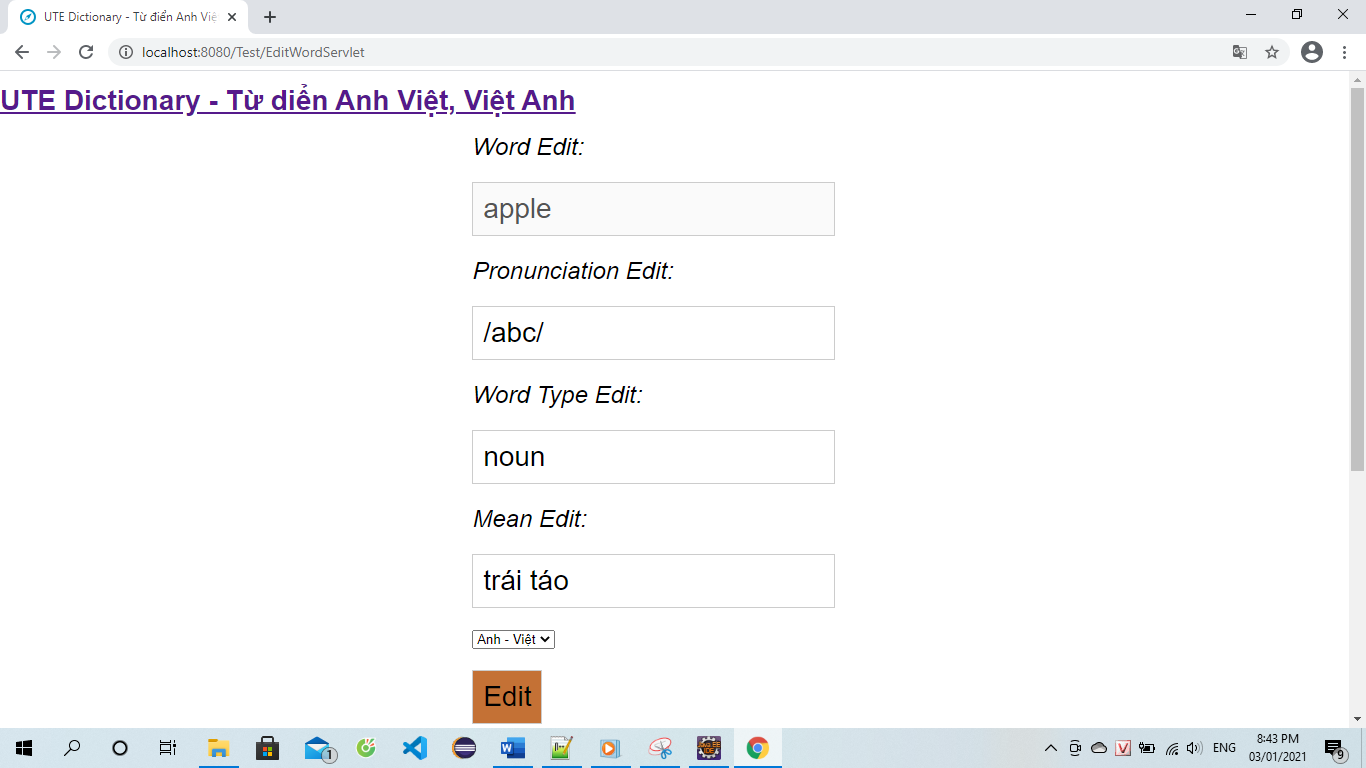
Sau khi tìm kiếm thì trang result.jsp, result2.jsp có tác dụng hiển thị từ cần tra:  
 Các thành phần chính của tranh hiển thị từ gần giống như trang home.jsp và home2.jsp tuy nhiên có thêm một số chức năng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jsp (HTML+ java)  + css+ javacript | Body | Thành phần | Tên | Mô tả |
| Jstl |  | Duyệt qua các nghĩa. |
| EL |  | Hiển thị các từ |

1.2. Trang quản lí các từ

Giao diện quản lí là giao diện chính dùng để cập nhật và quản lí tất cả các từ giao diện giúp người xem có thể thực hiện các thao tác thêm xoá sửa một từ trong bảo cùng với việc cung cấp các input nhất định để cho các tầng bên dưới xử lý:

Trang cho phép hiện và chỉnh sửa các từ :



Hình 1.2: Thiết kế trang web chỉnh sửa

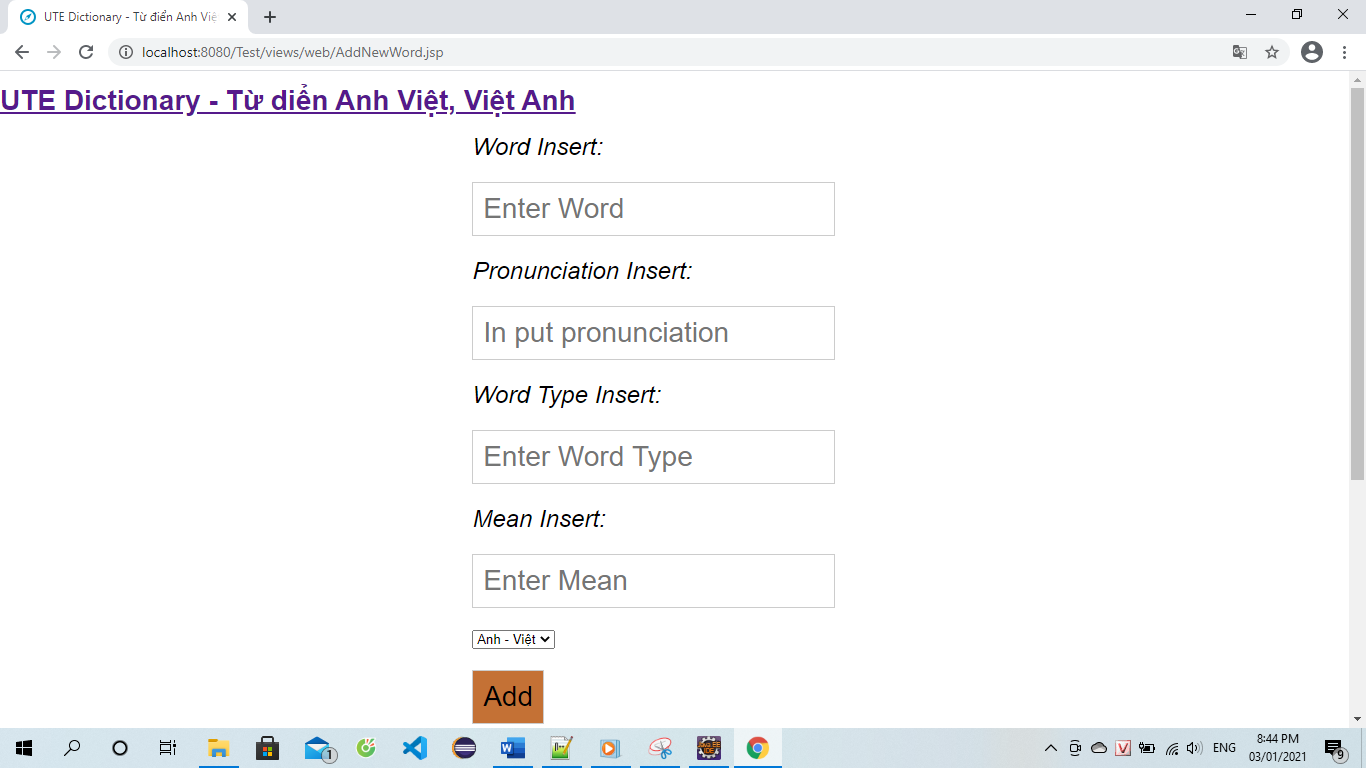
Trong phần chỉnh sửa từ vựng thì nhóm thực hiện sẽ có hai trang khác nhau cùng thực hiện chức năng là nhập từ chỉnh sửa. Hai trang này tương ứng với việc đỉnh chỉnh theo Anh-Việt hoặc điều chỉnh Việt-Anh (tức từ trong cây nhị phân của Anh-Việt và từ trong cây nhị phân Việt-Anh).

Các thành phần cấu trúc bao gồm:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jsp (HTML+ java)+ css+ javacript | Header | Thành phần | Tên | Mô tả |
| Class | logo | Class của css chứa file ảnh logo. |
| Thẻ a |  | Nằm trong logo chứa link dẫn về trang chủ. |
| Body | Textbox | Word Edit  Pronunciation edit  Wordtype edit  Mean edit | Phục vụ cho việc hiển thị từ cần chỉnh sửa(không thể thay đổi textbox này).  Phục vụ việc nhập cách đọc mới.  Phục vụ cách nhập loại từ mới  Phục vụ cho việc nhập nghĩa mới. |
| Select tag | searchOptions | Lựa chọn từ điển cần tra. |
| Submit | Edit | Đưa ra yêu cầu thực hiện việc chỉnh sửa. |
| Submit | Remove | Đưa ra yêu cầu thực hiện việc xoá từ. |
| Form |  | Tiến hành cung cấp dữ liệu để lớp serlvet thực hiện. |
| Thẻ a |  | Cho phép người dùng quay trở lại việc thêm từ mới. |
| Footer |  |  | Chứa các thông tin giới thiệu, đường link. |

Tương tự ở trang chỉnh sửa từ của Việt Anh vẫn lấy tương tự như trang chỉnh sửa từ của Anh việt.

1.3. Trang thêm một từ mới:



Hình 1.3: Trang thêm một từ mới

Trang web cho phép thực hiện thao tác thêm một một từ nào đó vào bên trong cây nhị phân. Ở đây nhóm thực hiện việc thêm một từ với điều kiện từ đó là từ tiếng Anh bao gồm cách đọc, loại từ, và nghĩa tiếng Việt của nó.

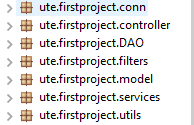
Các thành phần bao gồm:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jsp (HTML+ java)+ css+ javacript | Header | Thành phần | Tên | Mô tả |
| Class | logo | Class của css chứa file ảnh logo. |
| Thẻ a |  | Nằm trong logo chứa link dẫn về trang chủ. |
| Body | Textbox | Word Insert  Pronunciation edit  Wordtype edit  Mean edit | Phục vụ cho việc nhập từ mới.  Phục vụ việc nhập cách đọc cho từ mới.  Phục vụ cách nhập loại từ cho từ mới.  Phục vụ cho việc nhập nghĩa cho từ mới |
| Select tag | searchOptions | Lựa chọn từ điển cần tra(ở đây nhóm thực hiện chỉ cho thực hiện việc thêm từ Anh Việt) |
| Submit | Add | Đưa ra yêu cầu thực hiện việc thêm mới của từ vào cây và database. |
| Form |  | Tiến hành cung cấp dữ liệu để lớp serlvet thực hiện. |
| Thẻ a |  | Cho phép người dùng quay trở lại việc thêm từ mới. |
| Footer |  |  | Chứa các thông tin giới thiệu, đường link. |

1. Nội dung chính

2.1. Xây dựng mô hình MVC và 3-tier

Đồ án được đặt ra và xây dựng trên mô hình MVC kết hợp với mô hình 3-tier. Việc xây dựng này có ý nghĩa đặc biệt quan trọng trong việc thiết kế, bảo trì cũng như nâng cấp sản phẩm.



Hình 2.1: Mô hình MVC và 3-tier

2.2. Ngôn ngữ, hệ quản trị cơ sở dữ liệu và công nghệ được sử dụng:

Ngôn ngữ đang sử dụng: java, javascript.

Hệ quản trị cơ sở dữ liệu: MySQL.

Công nghệ sử dụng: JSP/Servlet, JQuery, GSON, JDBC.

2.3. Cấu trúc dữ liệu và giải thuật

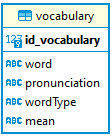
Sử dụng cây nhị phân tìm kiếm tự cân bằng. Một cây tìm kiếm nhị phân cân bằng có đặc điểm:

* Cây con bên trái của một nút có khóa (key) nhỏ hơn giá trị khóa của nút cha (của cây con này).
* Cây con bên phải của một nút có khóa lớn hơn giá trị khóa của nút cha (của cây con này).
* Tại mỗi nút chiều cao của hai cây con sai khác nhau không quá một.
* Hiệu quả là các phép chèn (insertion), và xóa (deletion) luôn chỉ tốn thời gian O(log n) trong cả trường hợp trung bình và trường hợp xấu nhất. Phép bổ sung và loại bỏ có thể cần đến việc tái cân bằng bằng một hoặc nhiều phép quay.

2.4. Ứng dụng giải quyết bài toán từ điển

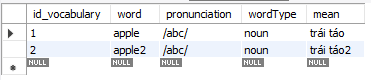
***- Phân tích và thiết kế hệ quản trị cơ sở dữ liệu:***

Sử dụng hệ quản trị cơ sở mysql làm cơ sở để phát triển ứng dụng web. Đầu tiên tạo một cơ sở dữ liệu chứa dữ liệu của trang web từ điển:



Chi tiết tổ chức từ trong cơ sở dữ liệu:

|  |  |  |
| --- | --- | --- |
| TT | Tên bảng | Mục đích của bảng trong chương trình |
| 1 | id\_vocabulary | Đánh dấu số thứ tự của một từ |
| 2 | word | Lưu trữ từ bằng tiếng anh |
| 3 | pronunciation | Lưu trữ phát âm của từ |
| 4 | wordType | Lưu trữ từ loại của từ |
| 5 | mean | Lưu trữ nghĩa của từ |



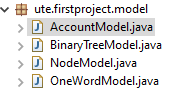
Hình 2.2: Mô tả cơ sở dữ liệu

***- Phân tích hình thành và xây dựng cấu trúc dữ liệu và giải thuật cho ứng dụng web:***

Công dụng chính của cây nhị phân tìm kiếm là việc tìm kiếm các Node trên cây một cách dễ dàng và tiết kiệm thời gian. Điều này rất phù hợp với thao tác tra từ (thao tác chính trên từ điển). Kèm theo các thao tác trên cây nhị phân tìm kiếm: như thêm một nút vào cây, xóa một nút trên cây…, các thao tác này cũng rất phù hợp với các thao tác cơ bản của phần mềm từ điển. Chính vì thế cấu trúc cây nhị phân tìm kiếm đã được ứng dụng vào chương trình một cách hiệu quả và triệt để.

***- Phân tích hình thành các gói phục vụ cho việc xây dụng ứng dụng web dựa trên mô hình MVC và 3-tier:***

Xây dựng tầng model phục vụ trong việc xây dựng mô hình cho một Node, một từ và một cây nhị phân cân bằng:



Xây dựng gói conn để thực hiện kết nối với hệ quản trị cơ sở dữ liệu:



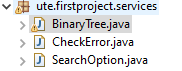
Để ràng buộc các kết nối bắt buộc phải do chính các servlet sử dụng nhằm đảm bảo tính bảo mật và toàn vẹn cơ sở dữ liệu việc tiếp theo là xây dựng một lớp lọc(filter) để ngăn các truy cập trái phép từ bên ngoài lớp servlet:



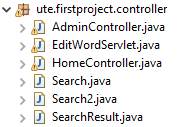
Thực hiện xây dựng tầng DAO nhằm mục đích thực hiện các thao tác trên hệ quản trị cơ sở dữ liệu:



Thực hiện xây dựng tầng service xử lý tất cả các logic cũng như tất cả các thao tác trên một cây nhị phân tìm kiếm cân bằng:



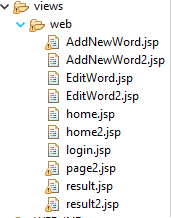
Thực hiện xây dựng tầng controller để luân chuyển dữ liệu giữa các tầng:



Thực hiện xây dựng tầng utils nhằm đảo bảo lưu trữ cây khi thực hiện giao thức http:



Thực hiện xây dựng nền cho trang web:

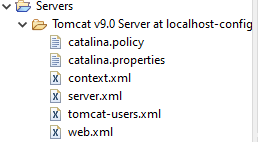


***- Mô hình hóa trang web trên server ảo:***

Apache Tomcat là một web server HTTP được phát triển bởi Apache Software Foundation, hỗ trợ mạnh cho ứng dụng Java thay vì website tĩnh. Do đó, nó có thể chạy nhiều bản Java chuyên biệt như Java Servlet, JavaServer Pages (JSP), Java EL, và WebSocket[1]. Thực hiện cấu hình trên server ảo bằng cách xử dụng pom.xml:



Một số cấu hình phụ để chạy được trên localhost:



2.5. Thiết kế lớp

|  |  |  |
| --- | --- | --- |
| TT | Tên lớp | Mục đích chính của lớp trong chương trình |
| 1 | ConnectionUtils.java | Mở, đóng, hoàn lại các tác vụ của cơ sở dữ liệu. |
| 2 | MySQLConnUtils.java | Kết nối với hệ quản trị cơ sở dữ liệu. |
| 3 | HomeController.java | Khởi tạo giá trị cho cây ban đầu và điều hướng lựa chọn tìm kiếm(Anh-Việt hoặc Việt-Anh). |
| 4 | Search.java | Sử dụng Gson để hiện gợi ý trong lúc tìm kiếm từ điển Anh-Việt. |
| 5 | Search2.java | Sử dụng Gson để hiện gợi ý trong lúc tìm kiếm từ điển Việt-Anh. |
| 6 | EditWordServlet.java | Thực hiện các thao tác điều khiển hiện thị trang web cho phép hiển thị và nhập các chỉnh sửa cần thiết,thực hiện thao tác chỉnh sửa sau khi nhận được dữ liệu đồng thời thực hiện việc xoá. Có các thao tác kiểm tra các input phù hợp cho việc thực hiện. |
| 7 | AdminController.java | Thực hiện việc thêm mới một từ vào bên trong cây nhị phân và thêm từ vào database. |
| 8 | IObjectDAO.java | Interface định nghĩa các hàm cho các thao tác tầng DAO. |
| 9 | Vocabulary.java | Thao tác với hệ quản trị cơ sở dữ liệu(lấy thông tin, thêm, xóa, chỉnh sửa). |
| 10 | JDBCFilter.java | Lọc các thao tác tới hệ quản trị cơ sở dữ liệu, chỉ cho phép lớp servlet thao tác với hệ quản trị cơ sở dữ liệu nhầm đảm bảo tính bảo mật và toạn vẹn thông tin. |
| 11 | BinaryTreeModel.java | Là một lớp javabean chứa thuộc tính và phương thức của cây. |
| 12 | NodeModel.java | Là một lớp javabean chứa thuộc tính và phương thức của một node. |
| 13 | OneWordModel.java | Là một lớp javabean chứa thuộc tính và phương thức của một từ. |
| 14 | Folder css | Chứa các lớp css để cấu hình một trang. |
| 15 | Folder js | Chứa các lớp js để cấu hình một trang. |
| 16 | home.jsp | Trang dùng để tìm kiếm từ Anh-Việt. |
| 17 | home2.jsp | Trang dùng để tìm kiếm từ Việt-Anh. |
| 18 | result.jsp | Trang dùng hiển thị từ khóa Anh-Việt đã tìm kiếm. |
| 19 | result2.jsp | Trang dùng hiển thị từ khóa Việt-Anh đã tìm kiếm. |
| 20 | index.jsp | Trỏ về HomeController.java để khởi tạo giá trị ban đầu cho cây. |
| 21 | pom.xml | Cấu hình web server, cài đặt thư viện, … |

2.6. Chi tiết các lớp

- ConnectionUtils.java

|  |  |
| --- | --- |
| Giải thích code | Phương thức được sử dụng để mở, đóng và hoàn lại các thao tác trên hệ quản trị cơ sở dữ liệu |
| Code trên java | //Mở kết nối  **public** **static** Connection getConnection() **throws** ClassNotFoundException, SQLException  {  **Return** MySQLConnUtils.*getMySQLConnection*();  } |
| //Đóng kết nối  **public** **static** **void** closeQuietly(Connection conn)  {  **try** {  conn.close();  } **catch** (Exception e) {  // **TODO**: handle exception  }  } |
| //Hoàn lại thao tác  **public** **static** **void** rollbackQuielty(Connection conn)  {  **try** {  conn.rollback();  } **catch** (Exception e) {  // **TODO**: handle exception  }  } |

* MySQLConnUtils.java

|  |  |
| --- | --- |
| Giải thích code | Các phương thức cấu hình truy cập hệ quản trị cơ sở dữ liệu |
| Code trên java | //Cấu hình mysql **public** **static** Connection getMySQLConnection() **throws** ClassNotFoundException, SQLException  {  String hostName="localhost";  String dbName="firstproject";  String userName="root";  String password="giakhang123";  **return** *getMySQLConnection*(hostName, dbName, userName, password);    } |
| //kết nối cơ sở dữ liệu  **public** **static** Connection getMySQLConnection(String hostName, String dbName, String userName, String password) **throws** ClassNotFoundException, SQLException  {  Class.*forName*("com.mysql.cj.jdbc.Driver");  String connectionURL= "jdbc:mysql://" + hostName + ":3306/" +dbName;    Connection conn= DriverManager.*getConnection*(connectionURL, userName, password);  **return** conn;  } |

* HomeController.java

|  |  |
| --- | --- |
| Giải thích code | Lấy session, khởi tạo giá trị ban đầu và điều hướng. |
| Code trên java | //Lấy session  BinaryTreeModel bTreeModel = MyUtils.*getBinaryTree*(request.getSession());  BinaryTreeModel bTreeModelVi = MyUtils.*getBinaryTreeVi*(request.getSession()); |
| //Khỏi tạo cây Anh-Việt  **if** (bTreeModel == **null**) {  List<OneWordModel> allWords = VocabularyDAO.*allWords*(MyUtils.*getStoredConnection*(request));  bTreeModel = **new** BinaryTreeModel();  BinaryTree bTree = **new** BinaryTree();  **for** (**int** i = 0; i < allWords.size(); i++) {  bTreeModel.setRoot(bTree.insertAVL(bTreeModel.getRoot(), allWords.get(i)));  }  bTree.find\_NLR(bTreeModel.getRoot());  MyUtils.*storeBinaryTree*(request.getSession(), bTreeModel);  } |
| // Khỏi tạo cây Việt-Anh  **if** (bTreeModelVi == **null**) {  List<OneWordModel> allWordsVi = VocabularyDAO.*allWordsVi*(MyUtils.*getStoredConnection*(request));  bTreeModelVi = **new** BinaryTreeModel();  BinaryTree bTree = **new** BinaryTree();  **for** (**int** i = 0; i < allWordsVi.size(); i++) {  bTreeModelVi.setRoot(bTree.insertAVL(bTreeModelVi.getRoot(), allWordsVi.get(i)));  }  bTree.find\_NLR(bTreeModelVi.getRoot());  MyUtils.*storeBinaryTreeVi*(request.getSession(), bTreeModelVi);  } |
| //Điều hướng đến các trang  String url = "";  **if** (request.getParameter("searchOptions") == **null**) {  url = "/views/web/home.jsp";  }  **else** **if** (Integer.*parseInt*(request.getParameter("searchOptions")) == 1) {  url = "/views/web/home.jsp";  }  **else** {  url = "/views/web/home2.jsp";  }      RequestDispatcher rd = getServletContext().getRequestDispatcher(url);  rd.forward(request, response); |

* VocabularyDAO

|  |  |
| --- | --- |
| Giải thích code | Thực hiện thao tác với hệ quản trị cơ sở dữ liệu |
| Code trên java | //Lấy tất cả các từ anh-việt  **public** **static** List<OneWordModel> allWords(Connection conn)  {  String sql = "Select \* from vocabulary";  **try** {  PreparedStatement pstm = conn.prepareStatement(sql);  ResultSet rs = pstm.executeQuery();  List<OneWordModel> allWords = **new** ArrayList<OneWordModel>();  **while** (rs.next()) {  OneWordModel newWord = **new** OneWordModel();  newWord.setWord(rs.getString(2));  newWord.setPronunciation(rs.getString(3));  newWord.setWordType(rs.getString(4));  newWord.setMean(rs.getString(5));  allWords.add(newWord);  }  **return** allWords;  } **catch** (SQLException e) {  e.printStackTrace();  }  **return** **null**;  } |
| //Lấy tất cả các từ Việt-Anh  **public** **static** List<OneWordModel> allWordsVi(Connection conn)  {  String sql = "Select \* from vocabulary";  **try** {  PreparedStatement pstm = conn.prepareStatement(sql);  ResultSet rs = pstm.executeQuery();  List<OneWordModel> allWords = **new** ArrayList<OneWordModel>();  **while** (rs.next()) {  OneWordModel newWord = **new** OneWordModel();  newWord.setMean(rs.getString(2));  newWord.setPronunciation(rs.getString(3));  newWord.setWordType(rs.getString(4));  newWord.setWord(rs.getString(5));  allWords.add(newWord);  }  **return** allWords;  } **catch** (SQLException e) {  e.printStackTrace();  }  **return** **null**;  } |
| //Thêm 1 từ vào cơ sở dữ liệu  **public** **boolean** add(Connection conn, Object obj) {  OneWordModel word = (OneWordModel)obj;  String sql= "Insert into vocabulary(word, pronunciation, wordType, mean) values(?,?,?,?)";  **try** {  PreparedStatement pstm = conn.prepareStatement(sql);  pstm.setString(1, word.getWord());  pstm.setString(2, word.getPronunciation());  pstm.setString(3, word.getWordType());  pstm.setString(4, word.getMean());  pstm.executeUpdate();  **return** **true**;  } **catch** (SQLException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    **return** **false**;  } |
| //Chỉnh sửa một từ trong cơ sở dữ liệu  **public** **boolean** edit(Connection conn, Object obj, String word) {  OneWordModel oneWord = **new** OneWordModel();  String sql = "Update vocabulary set pronunciation=?, wordType=?, mean=? Where word=?";  **try** {  PreparedStatement pstm = conn.prepareStatement(sql);  pstm.setString(1, oneWord.getPronunciation());  pstm.setString(2, oneWord.getWordType());  pstm.setString(3, oneWord.getMean());  pstm.setString(4, oneWord.getWord());  pstm.executeUpdate();  **return** **true**;  } **catch** (SQLException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    **return** **false**;  } |
| //Xóa một từ trong cơ sở dữ liệu  **public** **boolean** delete(Connection conn, String word) {  OneWordModel oneWord = **new** OneWordModel();  String sql = "Delete \* From vocabulary Where word=?";  **try** {  PreparedStatement pstm = conn.prepareStatement(sql);  pstm.setString(1, oneWord.getWord());  pstm.executeUpdate();  **return** **true**;  } **catch** (SQLException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    **return** **false**;  } |
| //Tìm kiếm một từ trong cơ sở dữ liệu  **public** List<String> search(Connection conn, String keyword)  {  List<String> names = **new** ArrayList<String>();  List<OneWordModel> allWords = *allWords*(conn);  **for** (OneWordModel oneWord : allWords) {  **if** (oneWord.getWord().toLowerCase().contains(keyword.toLowerCase())) {  names.add(oneWord.getWord());  }  }  **return** names;  } |

* BinaryTree.java

|  |  |
| --- | --- |
| Giải thích code | Tìm node thay thế. Truyền vào node bên phải của node cần xóa và trả về node trái nhất của node bên phải. |
| Code trên java | **public** NodeModel minValueNode(NodeModel node) {  NodeModel current = node;  /\* loop down to find the leftmost leaf \*/  **while** (current.getpLeft() != **null**)  current = current.getpLeft();  **return** current;  } |
| Giải thích code | Trong thuật toán này ta xem việc xóa nút trong trường hợp là nút lá hoặc có 1 nút con là như nhau. Trong trường hợp cây có 1 nút con thì nút cha của x sẽ nối với con duy nhất của x, còn nếu x là nút lá thì nói nút cha của x với 1 trong 2 nút con của x đều có giá trị là null. Trường hợp nút có 2 con, chúng ta sẽ thay thế x bằng nút nhỏ nhất bên cây con phải (là phần tử cực trái trên cây con phải). Sau khi xóa node thực hiện cân bằng lại cây. |
| Code trên java | **public** NodeModel deleteNode(NodeModel node, String key)  {  **if** (node == **null**) {  **return** node;  }  **else** {  **if** (key.compareTo(node.getData().getWord())<0) {  node.setpLeft(deleteNode(node.getpLeft(), key));  }  **else** **if** (key.compareTo(node.getData().getWord())>0) {  node.setpRight(deleteNode(node.getpRight(), key));  }  **else** {  **if**( (node.getpLeft() == **null**) || (node.getpRight() == **null**) ) {  NodeModel temp;  **if** (node.getpLeft() != **null**)  temp = node.getpLeft();  **else**  temp = node.getpRight();  // No child case  **if**(temp == **null**) {  temp = node;  node = **null**;  }  **else** // One child case  node = temp; // Copy the contents of the non-empty child  temp = **null**;  }  **else** {  // node with two children: Get the inorder successor (smallest  // in the right subtree)  NodeModel temp = minValueNode(node.getpRight());  // Copy the inorder successor's data to this node  node.setData(temp.getData());  // Delete the inorder successor  node.setpRight(deleteNode(node.getpRight(), key)) ;  }  }  }  **if** (node == **null**) {  **return** node;  }  node.setHeight(1+max(height(node.getpLeft()), height(node.getpRight())));  // STEP 3: GET THE BALANCE FACTOR OF THIS NODE (to check whether  // this node became unbalanced)  **int** balance = getBalance(node);  // If this node becomes unbalanced, then there are 4 cases  // Left Left Case  **if** (balance > 1 && getBalance(node.getpLeft()) >= 0)  **return** rightRotate(node);  // Left Right Case  **if** (balance > 1 && getBalance(node.getpLeft()) < 0) {  node.setpLeft(leftRotate(node.getpLeft()));  **return** rightRotate(node);  }  // Right Right Case  **if** (balance < -1 && getBalance(node.getpRight()) <= 0)  **return** leftRotate(node);  // Right Left Case  **if** (balance < -1 && getBalance(node.getpRight()) > 0) {  node.setpRight(rightRotate(node.getpRight()));  **return** leftRotate(node);  }  **return** node;  } |
| Giải thích code | Tìm kiếm một node. |
| Code trên java | **public** NodeModel Search(NodeModel node, String key)  {  NodeModel p = node;  **while** (p!=**null**) {  **if** (key.compareTo(p.getData().getWord())==0) {  **return** p;  }  **else** **if** (key.compareTo(p.getData().getWord())<1) {  p=p.getpLeft();  }  **else** {  p=p.getpRight();  }    }    **return** **null**;    } |
| Giải thích code | Tìm độ cao của một node. |
| Code trên java | **public** **int** height(NodeModel N)  {  **if** (N == **null**) {  **return** 0;  }    **return** N.getHeight();  } |
| Giải thích code | Trả về chiều cao của node lớn hơn. |
| Code trên java | **public** **int** max(**int** a, **int** b)  {  **return** (a>b) ? a : b;  } |
| Giải thích code | Xoay bên phải. |
| Code trên java | **public** NodeModel rightRotate(NodeModel y)  {  NodeModel x = y.getpLeft();  NodeModel temp = x.getpRight();    x.setpRight(y);  y.setpLeft(temp);    y.setHeight(max(height(y.getpLeft()), height(y.getpRight()))+1);  x.setHeight(max(height(x.getpLeft()), height(x.getpRight()))+1);    **return** x;  } |
| Giải thích code | Xoay bên trái. |
| Code trên java | **public** NodeModel leftRotate(NodeModel x)  {  NodeModel y =x.getpRight();  NodeModel temp = y.getpLeft();    y.setpLeft(x);  x.setpRight(temp);    x.setHeight(max(height(x.getpLeft()), height(x.getpRight()))+1);  y.setHeight(max(height(y.getpLeft()), height(y.getpRight()))+1);    **return** y;      } |
| Giải thích code | Trả về kết quả chiều cao của cây con bên trái trừ cho cây con bên phải. |
| Code trên java | **public** **int** getBalance(NodeModel N)  {  **if** (N == **null**) {  **return** 0;  }    **return** height(N.getpLeft())-height(N.getpRight());  } |
| Giải thích code | Đối với cây nhị phân tìm kiếm cân bằng, sau khi thực hiện chèn một nút vào cây, ta kiểm tra xem cây có bị mất cân bằng hay không. Nếu có, ta thực hiện thao tác xoay trái hoặc xoay phải cây (tùy theo trường hợp mất cân bằng) để đưa cây về trạng thái cân bằng. |
| Code trên java | **public** NodeModel insertAVL(NodeModel node, OneWordModel word)  {  **if** (node == **null**) {  **return** (**new** NodeModel(word));  }    **if** (word.getWord().compareTo(node.getData().getWord())<0) {  node.setpLeft(insertAVL(node.getpLeft(), word));  }  **else** **if** (word.getWord().compareTo(node.getData().getWord())>0) {  node.setpRight(insertAVL(node.getpRight(), word));  }  **else** {  **return** node;  }      node.setHeight(1+max(height(node.getpLeft()), height(node.getpRight())));    **int** balance = getBalance(node);    //4 case  // Left Left  **if** (balance > 1 && word.getWord().compareTo(node.getpLeft().getData().getWord())<0) {  **return** rightRotate(node);  }    // Right Right  **if** (balance <-1 && word.getWord().compareTo(node.getpRight().getData().getWord())>0) {  **return** leftRotate(node);  }    //Left Right  **if** (balance > 1 && word.getWord().compareTo(node.getpLeft().getData().getWord())>0) {  node.setpLeft(leftRotate(node.getpLeft()));  **return** rightRotate(node);  }  // Right Left  **if** (balance<-1 && word.getWord().compareTo(node.getpRight().getData().getWord())<0) {  node.setpRight(rightRotate(node.getpRight()));  **return** leftRotate(node);  }    **return** node;  } |

* Pom.xml

|  |  |
| --- | --- |
| Giải thích code | Sử dụng để cấu hình chạy server |
| Code trên java | <project xmlns=*"http://maven.apache.org/POM/4.0.0"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>ute.firstproject</groupId>  <artifactId>dictionary</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.source>1.8</maven.compiler.source>  <maven.compiler.target>1.8</maven.compiler.target>  </properties>  <dependencies>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <version>2.0</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5</version>  </dependency>  <dependency>  <groupId>javax.servlet.jsp.jstl</groupId>  <artifactId>jstl-api</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.21</version>  </dependency>    <dependency>  <groupId>com.google.code.gson</groupId>  <artifactId>gson</artifactId>  <version>2.8.5</version>  </dependency>  </dependencies>  <build>  <finalName>dictionary</finalName>  <pluginManagement>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.6.1</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <plugin>  <artifactId>maven-clean-plugin</artifactId>  <version>3.1.0</version>  </plugin>  <plugin>  <artifactId>maven-resources-plugin</artifactId>  <version>3.0.2</version>  </plugin>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.8.0</version>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.1</version>  </plugin>  <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>3.2.2</version>  </plugin>  <plugin>  <artifactId>maven-install-plugin</artifactId>  <version>2.5.2</version>  </plugin>  <plugin>  <artifactId>maven-deploy-plugin</artifactId>  <version>2.8.2</version>  </plugin>  </plugins>  </pluginManagement>  </build>  </project> |

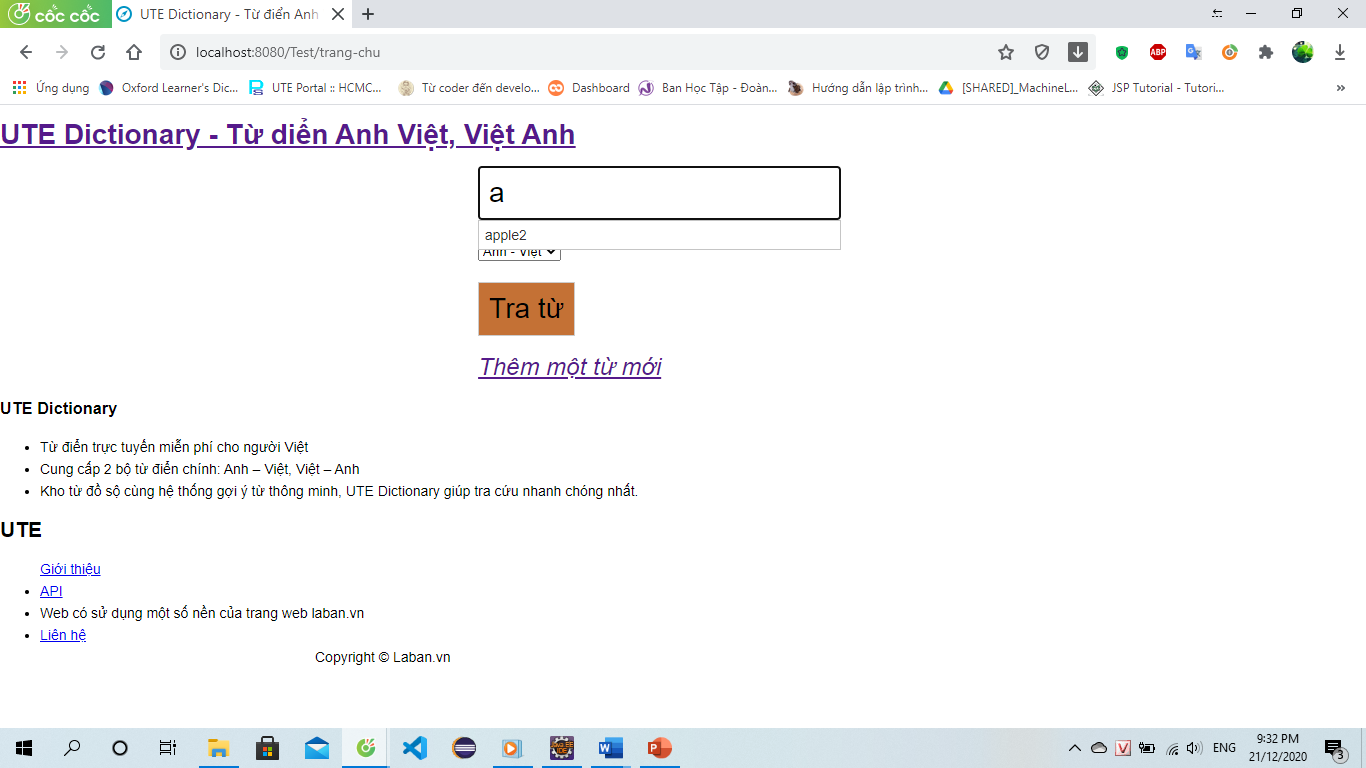
1. Cài đặt và kiểm thử

Dữ liệu được lưu trữ bên tron database để tiến hành việc tạo lập dữ liệu ban đầu cho trang web. Mỗi khi truy cập vào trang web lần đầu tiên, dữ liệu từ database sẽ được load lên và tiến hành lưu trữ vào 2 cây nhị phân là cây nhị phân Việt Anh và cây nhị phân Anh Việt sau trang web sẽ được chuyển đến trang search để thực hiện các thao tác sau đó.

Trong quá trình thực hiện nhóm đã thực hiện việc kiểm thử những chức năng sau đây:

3.1. Tìm kiếm từ

Sau khi nhập từ cần tìm kiếm vào thanh textbox tìm kiếm và ấn tra từ. Lúc này dựa vào ta đang sử dụng ở cây nhị phân nào thì chương trình web sẽ trả về kết quả được hiện ở phía bên dưới. Ví dụ ta ấn từ university, khi ta ấn các kí tự như u,n,i thì chương trình sẽ hiện cho ta các đề xuất tra từ. Sau đó sẽ hiện ra thông tin của từ đó. Như ở đây ta thấy từ điển sẽ đưa cho ta từ vừa tra, cách đọc, loại từ, cách phát âm.



Hình 3.1.1: Gợi ý tra từ



Hình 3.1.2: Trang kết quả tra từ

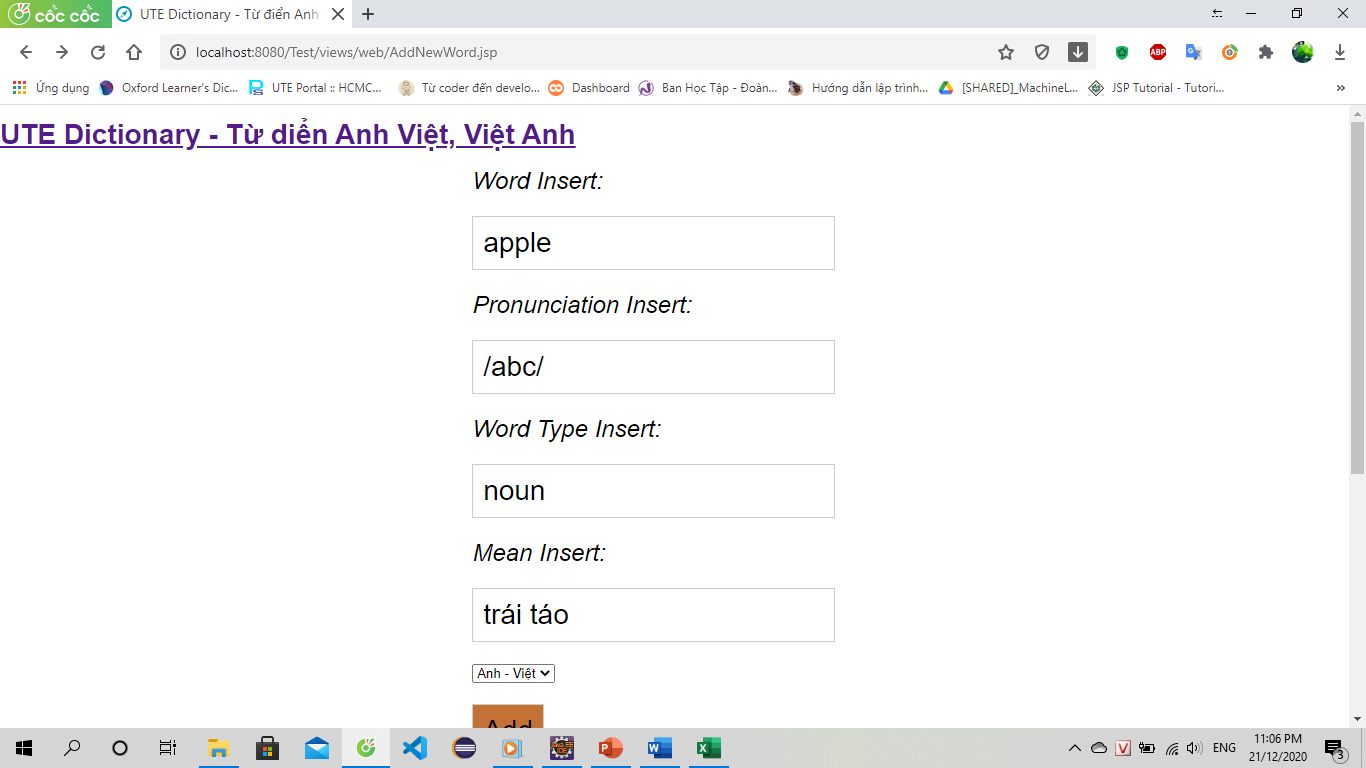
Trong trường hợp chưa nhập từ hoặc nhập sai từ thì chuong trình sẽ hiện lên dòng chữ: “Xin lỗi! Không thể tìm thấy từ này”.



Hình 3.1.3: Trang không tìm thấy từ cần tra

3.2. Cập nhật từ

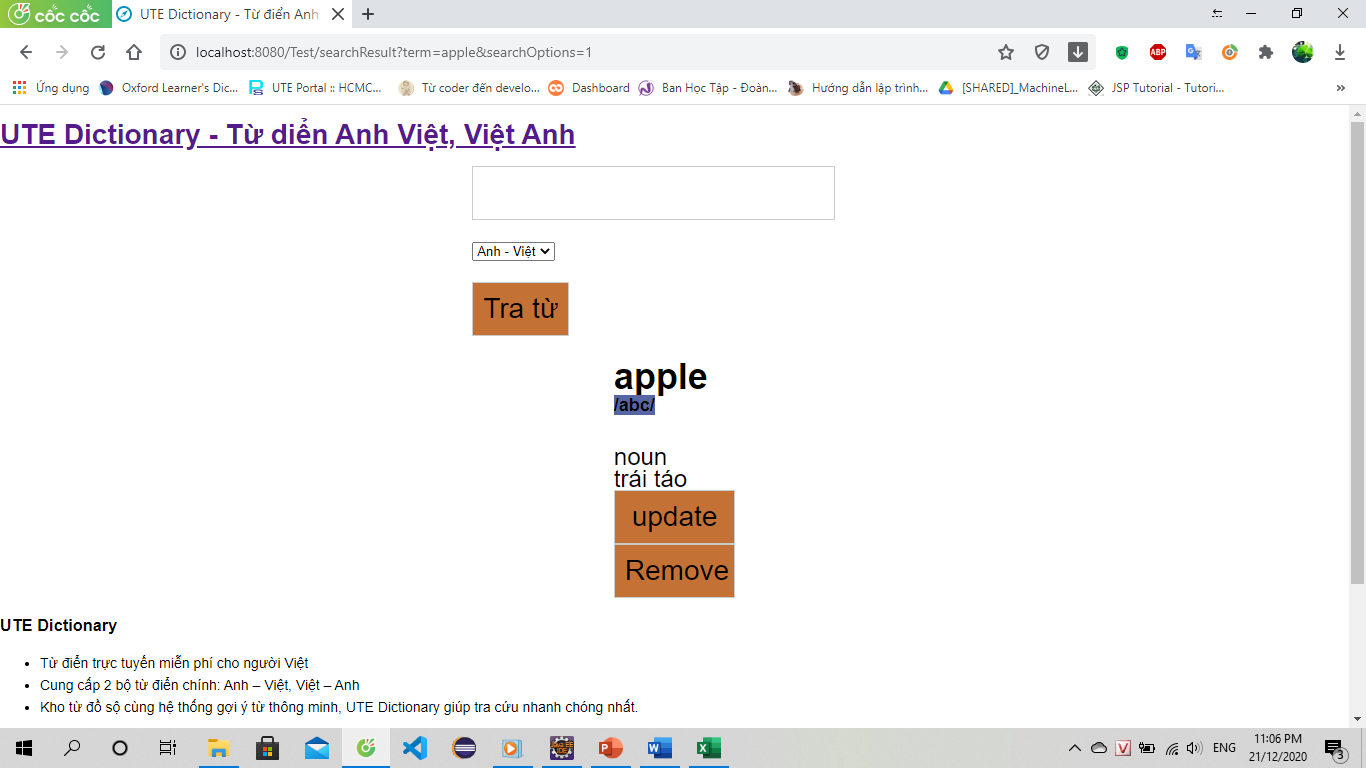
Đầu tiên ta sẽ tiến hành nhập từ apple vào thanh tìm kiếm. Trang web sẽ hiện ra kết quả của từ đó trong trường hợp người dùng muốn update từ thì tiến hành ấn nút update để trang web chuyển hướng đến trang update.



Hình 3.2.1: Trang thêm từ mới

Tại đây sẽ xuất hiện các từ là các thanh text box cho phép thực hiện việc điều chỉnh cho từ này. Ở đây trang web còn lưu những biến hidden như WordSearch để giúp lớp control thực hiện thuận tiện.

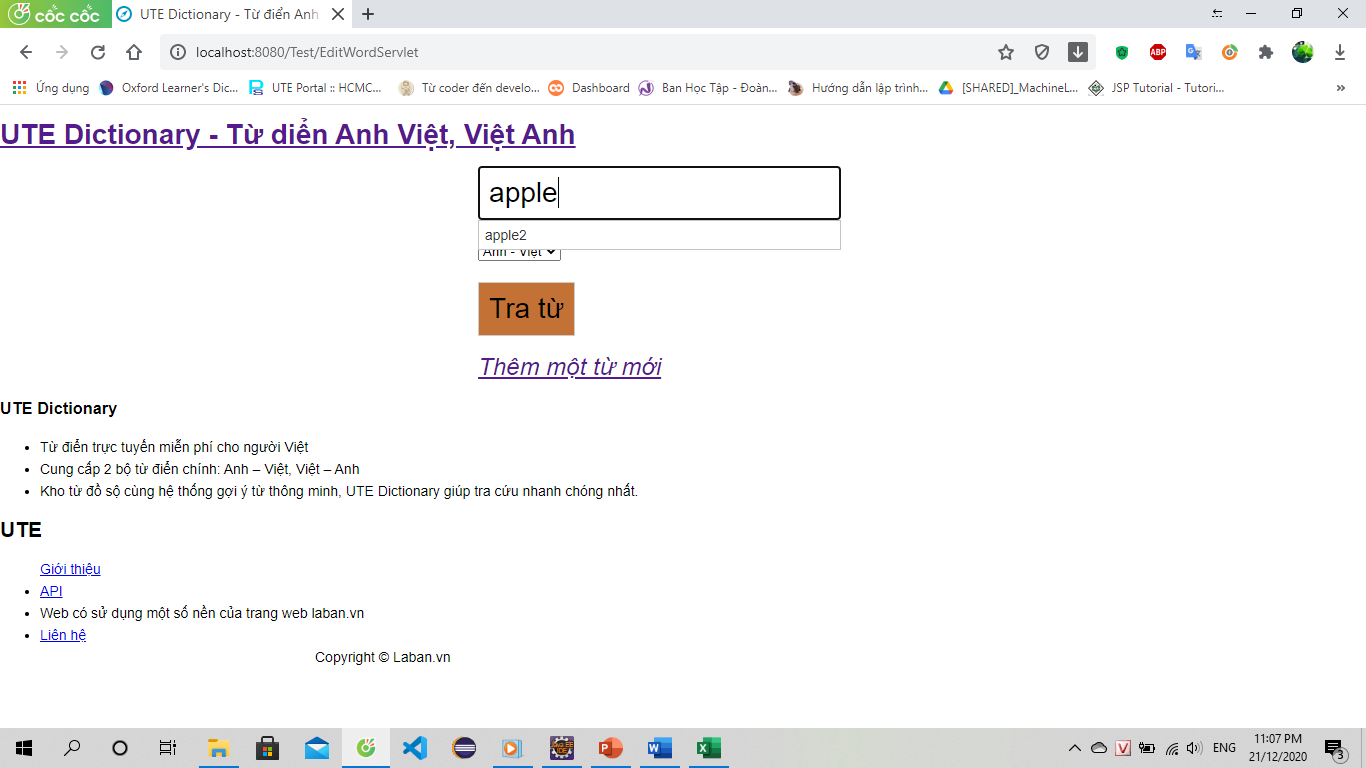
Sau khi ta thực hiện thay đổi ví dụ như chuyển WordType thành noun và phần pronunciation chuyển thành /'æpl/.Tiếp đó ấn button edit thì từ khoá sẽ được cập nhật lại ở cả hai cây nhị phân. Đã được cập nhật ở cây nhị phân Việt Anh.



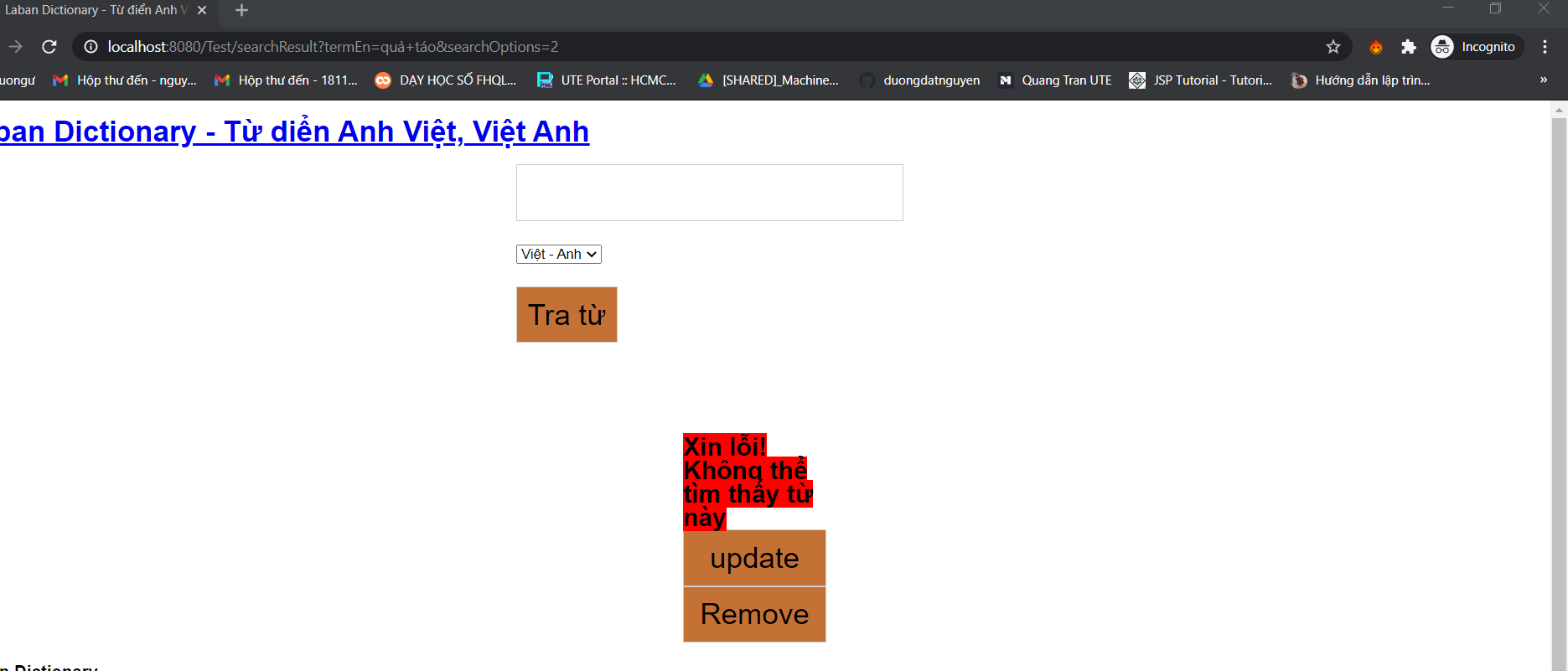
Hình 3.2.2: Tìm kiếm lại từ đã thêm

3.3. Xoá từ

Ở hình 3.2 khi ta ấn button Remove thì từ sẽ tiến hành xoá khỏi cây nhị phân và xoá khỏi database điều này được thực hiện thông qua một biến hidden được truyền word của từ đang hiện lên vào và tiến hành xoá nó đi. Cuối cùng nếu thực hiện xong thì trang web sẽ được trả về trang home tương ứng với dữ liệu ban đầu. Ở đây nhóm thực hiện tiến hành xoá dữ liệu của quả táo. Sau khi tìm kiếm thì ta không còn thấy dữ liệu của quả táo đâu nữa cùng với đó là dữ liệu của apple cũng biến mất.



Hình 3.3: Tìm kiếm lại từ đã xóa



Hình 3.3: Tìm kiếm lại từ đã xóa

Đồng thời ta có thể sử dụng button remove có hiện thị trong khi edit word, button này có tính năng tương tự như button mà nhóm thực hiện đã demo trên đây.

## III.Phân công công việc

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STT** | **DANH SÁCH CÔNG VIỆC** | **Ghi chú** | **Nguyễn Lâm Gia Khang** | **Nguyễn Dương Đạt** | **Ngày bắt đầu (dự kiến)** | **Ngày kết thúc (dự kiến)** | **Ngày bắt đầu (thực tế)** | **Ngày kết thúc (thực tế)** |
| 1 | Tìm hiểu JSP/Servlet + MySQL |  | x | x | 22/10/2020 | 07/11/2020 | 22/10/2020 | 10/11/2020 |
| 2 | Thiết kế trang chính cho từ điển |  | x | x | 08/11/2020 | 14/11/2020 | 11/11/2020 | 14/11/2020 |
| 3 | Thiết kế database cho từ điển |  | x | x | 08/11/2020 | 14/11/2020 | 08/11/2020 | 14/11/2020 |
| 4 | Xây dựng các class |  | x | x | 15/11/2020 | 10/12/2020 | 15/11/2020 | 18/12/2020 |
| 5 | Viết báo cáo + làm ppt |  | x | x | 19/12/2020 | 20/12/2020 | 19/12/2020 | 21/12/2020 |
| 6 | Demo lần 1 |  | x | x | 22/12/2020 | 22/12/2020 |  |  |

## IV.Kết luận

4.1. Đánh giá mức độ hoàn thành

Nhóm thực hiện tự đánh giá rằng nhóm đã cơ bản hoàn thành các yêu cầu của đồ án. Trong đồ án này, nhóm thực hiện đã củng cố lại các kiến thức về cây nhị phân tìm kiếm, cách tổ chức dữ liệu trên cây và các thao tác trên cây như tìm kiếm, thêm và xóa nút trên cây. Áp dụng vào trang web từ điển để thực hiện các chức năng tra từ, chỉnh sửa, thêm từ mới và xóa từ.Ngoài ra nhóm cũng còn vận dụng các kiến thức cơ bản của các môn học đã trải qua như lập trình hướng đối tượng, vận dụng môn lập trình web để tạo lập một trang web ổn định, kết hợp với database kết nối xử lý với cơ sở dữ liệu. Ngoài ra nhóm thực hiện còn tích luỹ thêm các kiến thức mới đáp án cho nhu cầu làm việc sau này. Thông qua đồ án nhóm thực hiện đã khắc phục được những vấn đề lớn tích luỹ những kinh nghiệm thực tiễn trong việc tạo lập một trang web.

Tuy nhiên đây chỉ là chương trình web mô phỏng và giúp tổng hợp kiến thức nên web này vẫn chương thể được ứng dụng vào trong thực tế nhiều. Chương trình chỉ là một web đơn giản cần nhiều cải tiến về giao diện cũng như các chức năng hơn.

4.2. Khó khăn

Trong quá trình làm việc nhóm chưa thể giải quyết vấn đề như việc lưu trữ một từ có nhiều nghĩa cũng như việc lưu trữ file ghi âm cho từ đã tra. Cùng với giao diện còn ở mức đơn sơ do nhóm chỉ mới vừa tiếp xúc với việc lập trình web gần đây.Đồng thời việc ứng dụng công nghệ mới tương đối khó khăn bằng chứng là việc thanh tiên đoán khi tìm kiếm bằng tiếng việc còn xảy ra lỗi kiểu chữ.Cùng với những khó khăn trong việc tiếp xúc với những IDE mới những phần mềm tương đối cũ như tomcat đã làm nhóm tốn quá nhiều thời gian trong việc chỉnh sửa phù hợp. Với những áp lực về thời gian nhóm đã không kịp thực hiện thêm các yêu cầu khác cũng như ứng dụng nhưng công nghệ tốt hơn cho web.

4.3. Hướng phát triển

Trong thời gian tiếp theo, nhóm thực hiện sẽ cố gắng cải thiện web với giao diện cũng như những các tính năng mới hơn, đáp ứng đầy đủ yêu cầu của người dùng hơn.

Các vấn đề còn lại của trang web, nhóm sẽ tiến hành tìm kiếm tài liệu để học hỏi nhiều hơn để tích luỹ kiến thức về việc tạo lập một trang web hơn. Song song với việc trên, nhóm thực hiện sẽ tiến hành kiểm tra lại trang web và tìm cách nâng cấp để làm giảm đi những vấn đề còn hạn chế của phần mềm.

Về phần dữ liệu của từ điển, nhóm thực hiện sẽ cố gắng lấy những dữ liệu lớn của các từ điển trực tuyến trên thế giới về để làm cho web càng đa dạng về từ hơn. Đồng thời xây dựng lại database cho web một cách đầy đủ hơn.

Tiếp theo, nhóm thực hiện sẽ thêm vào đó các tính năng cho phép người truy cập có thể học được từ vựng, cùng với đó là rèn luyện trí nhớ cho các từ đã học qua. Xây dựng một web hoc tập trực tuyến thu nhỏ bên trong web từ điển này. Từ đó ứng dụng trang web và đưa trang web lên internet để tiến hành áp dụng vào thực tế.

## V. Phụ lục

* Search.java

|  |  |
| --- | --- |
| Giải thích code | Sử dụng thư viện Gson để hiện gợi ý trên từ điển Anh – Việt |
| Code trên java | Gson gson = **new** Gson();  String term = request.getParameter("term");  // System.out.print(term);  // String termEn = request.getParameter("termEn");  // System.out.print(termEn);  SearchOption searchOption = **new** SearchOption();  PrintWriter out = response.getWriter();  out.print(gson.toJson(searchOption.search(MyUtils.*getBinaryTree*(request.getSession()).getRoot(), term)));  out.flush();  out.close(); |

* Search2.java

|  |  |
| --- | --- |
| Giải thích code | Sử dụng thư viện Gson để hiện gợi ý trên từ điển Anh – Việt |
| Code trên java | Gson gson = **new** Gson();  String term = request.getParameter("term");  System.***out***.print(term);  SearchOption searchOption = **new** SearchOption();  PrintWriter out = response.getWriter();  out.print(gson.toJson(searchOption.search(MyUtils.*getBinaryTreeVi*(request.getSession()).getRoot(), term)));  out.flush();  out.close(); |

* SearchResult.java

|  |  |
| --- | --- |
| Giải thích code | Nhận vào giá trị searchOptions(1 là từ diển Anh-Việt và 2 là từ diển Việt-Anh). Tìm kiếm trong cây cân bằng và in ra kết quả. |
| Code trên java | //Nhận vào giá trị searchOption và khởi tạo các biến ban đầu  String searchOptions = request.getParameter("searchOptions");  BinaryTree bTree = **new** BinaryTree();  String error = "";  String url = ""; |
| //Tìm kiếm trên từ điển Anh-Việt  **if** (Integer.*parseInt*(searchOptions) == 1) {  **if** (request.getParameter("term") == **null**) {  error = "Bạn vui lòng điền một từ";  request.setAttribute("error", error);  }  **else** {  String word = request.getParameter("term");  System.***out***.print(word);  NodeModel node = bTree.Search(MyUtils.*getBinaryTree*(request.getSession()).getRoot(), word);  **if** (node == **null**) {  error = "Xin lỗi! Không thể tìm thấy từ này";  request.setAttribute("error", error);  }  **else** {  request.setAttribute("result", node);  }  }  url = "/views/web/result.jsp";  } |
| //Tìm kiếm trên từ điển Việt-Anh  **else** {  **if** (request.getParameter("termEn") == **null**) {  error = "Bạn vui lòng điền một từ";  request.setAttribute("error", error);  }  **else** {  String word = request.getParameter("termEn");  NodeModel node = bTree.Search(MyUtils.*getBinaryTreeVi*(request.getSession()).getRoot(), word);  **if** (node == **null**) {  error = "Xin lỗi! Không thể tìm thấy từ này";  request.setAttribute("error", error);  }  **else** {  request.setAttribute("result", node);  }  }    url = "/views/web/result2.jsp";  } |
| //Điều hướng  RequestDispatcher rd = getServletContext().getRequestDispatcher(url);  rd.forward(request, response); |

* IObjectDAO

|  |  |
| --- | --- |
| Giải thích code | Interface là một kỹ thuật để thu được tính trừu tượng hoàn toàn và đa kế thừa trong Java. Thực hiện trong việc định nghĩa các phương thức cho lớp DAO: thêm, xóa, chỉnh sửa. |
| Code trên java | **public** **boolean** add(Connection conn, Object obj);  **public** **boolean** edit(Connection conn, Object obj, String word);  **public** **boolean** delete(Connection conn, String word); |

* JDBCFilter.java

|  |  |
| --- | --- |
| Giải thích code | Nó tập hợp 1 map bao gồm các lớp servlet sau đó mỗi khi servlet truy cập đến database thì kết nối sẽ được mở. Lớp này có tác dụng chặn các truy cập không hợp pháp như file ảnh, jsp, … |
| Code trên java | System.***out***.print("JDBC Filter");  String servletPath=request.getServletPath();  String pathInfo = request.getPathInfo();  String urlPattern = servletPath;  **if** (pathInfo !=**null**) {  urlPattern = servletPath+"/\*";  }    Map<String, ?**extends** ServletRegistration> servletRegistration = request.getServletContext()  .getServletRegistrations();  Collection<? **extends** ServletRegistration> values = servletRegistration.values();  **for** (ServletRegistration sr : values) {  Collection<String> mappings = sr.getMappings();  **if** (mappings.contains(urlPattern)) {  **return** **true**;  }  }  **return** **false**; |

* BinaryTreeModel.java

|  |  |
| --- | --- |
| Giải thích code | Đây là lớp chứa các thuộc tính và phương thức getter và setter của một cây. |
| Code trên java | **private** NodeModel root;    **public** NodeModel getRoot() {  **return** root;  }  **public** **void** setRoot(NodeModel root) {  **this**.root = root;  }    **public** BinaryTreeModel()  {  root = **null**;  } |

* NodeModel.java

|  |  |
| --- | --- |
| Giải thích code | Đây là lớp chứa các thuộc tính và phương thức getter và setter của một Node bao gồm: Node trái, Node phải, dữ liệu và chiều cao. |
| Code trên java | **private** OneWordModel data;  **private** NodeModel pLeft;  **private** NodeModel pRight;  **private** **int** height;        **public** OneWordModel getData() {  **return** data;  }  **public** **void** setData(OneWordModel data) {  **this**.data = data;  }  **public** NodeModel getpLeft() {  **return** pLeft;  }  **public** **void** setpLeft(NodeModel pLeft) {  **this**.pLeft = pLeft;  }  **public** NodeModel getpRight() {  **return** pRight;  }  **public** **void** setpRight(NodeModel pRight) {  **this**.pRight = pRight;  }  **public** **int** getHeight() {  **return** height;  }  **public** **void** setHeight(**int** height) {  **this**.height = height;  }    **public** NodeModel()  {    }    **public** NodeModel(OneWordModel word)  {  // data = word;  // height = 1;  // pLeft = null;  // pRight = null;    setData(word);  setHeight(1);  setpLeft(**null**);  setpRight(**null**);  }    **public** **void** setNode(NodeModel node) {  **this**.data = node.data;  **this**.pLeft=node.pLeft;  **this**.pRight=node.pRight;  **this**.height=node.height;  } |

* OneWordModel.java

|  |  |
| --- | --- |
| Giải thích code | Đây là phương thức chứa thuộc tính và phương thức getter và setter của một từ bao gồm: từ, phát âm, loại từ và nghĩa. |
| Code trên java | **private** String word;  **private** String pronunciation;  **private** String wordType;  **private** String mean;      **public** OneWordModel()  {    }    **public** String getWord() {  **return** word;  }  **public** **void** setWord(String word) {  **this**.word = word;  }  **public** String getPronunciation() {  **return** pronunciation;  }  **public** **void** setPronunciation(String pronunciation) {  **this**.pronunciation = pronunciation;  }  **public** String getWordType() {  **return** wordType;  }  **public** **void** setWordType(String wordType) {  **this**.wordType = wordType;  }  **public** String getMean() {  **return** mean;  }  **public** **void** setMean(String mean) {  **this**.mean = mean;  } |

* AdminController.java

|  |  |
| --- | --- |
| Giải thích code | Đâu tiên ta định dạng characterEncoding để ta có thể lấy từ tiếng việt. Tiếp theo ta lấy các dữ liệu mà request trả về như là action (hành động thực hiện), lấy dữ liệu của hai cây nhị phân Anh Việt và Việt Anh mà ta đã tiến hành lưu nó ở phần trước. |
| Code trên java | request.setCharacterEncoding("UTF-8");  String action=request.getParameter("action");  String searchOptions = request.getParameter("searchOptions");  BinaryTreeModel treeModel=MyUtils.*getBinaryTree*(request.getSession());  BinaryTreeModel treeModelVA=MyUtils.*getBinaryTreeForViet\_Anh*(request.getSession());  String url=""; |
| Giải thích code | Trong trường hợp hành động ta nhận được là thêm từ và thuộc dạng là thêm từ Anh Việt. Ta tiến hành lấy các dữ liệu như từ, nghĩa, loại từ, cách phát âm của từ mới thông qua hàm getParameter để lấy dữ liệu từ trang nhập.  Ở đây nhóm thực hiện có một model CheckError dùng để kiểm tra xem có xuất hiện lỗi từ phía người nhập hay không. Trong trường hợp có xuất hiện lỗi thì ta sẽ trả về trang thêm từ và kèm theo các lỗi và dữ liệu đã nhập cho người dùng kiểm tra. |
| Code trên java | **if**(action.equals("Add")&&searchOptions.equals("1"))  {    String word=request.getParameter("WordAdd").trim();  String pronunciation=request.getParameter("Pronunciation").trim();  String wordtype=request.getParameter("WordType").trim();  String mean=request.getParameter("Mean").trim();  CheckError checkError=**new** CheckError();  **boolean** error=CheckErrorInput(mean, pronunciation, word, wordtype, checkError, treeModel);  System.***out***.println("Have error"+error);  **if**(error)  {  request.setAttribute("checkError", checkError);  request.setAttribute("word", word);  request.setAttribute("wordtype", wordtype);  request.setAttribute("pronunciation", pronunciation);  request.setAttribute("mean", mean);  url="/views/web/AddNewWord.jsp";  } |
| Giải thích code | Đầu tiên ta tiến hành tạo một node model và thêm vào cây nhị phân Anh Việt(ở đây là TreeModel mà ta đã khai báo phía trên),sau đó ta ta tiếng anh lưu trữ cây đó điển tiện cho việc thực hiện sau này. Sau đó ta tiến hành tạo một mode Word để lưu trữ vào database và ta lưu trữ vào database.  Tiếp theo ta tiến hành thêm từ mới vào cây nhị phân Việt Anh còn lại và lưu trữ cây nhị phân đó và gán một url để trả lại trang ban đầu. |
| Code trên java | InsertModelToBinary(mean, pronunciation, word, wordtype, searchOptions, treeModel);  MyUtils.*storeBinaryTree*(request.getSession(), treeModel);  //NodeModel nodeModel=new NodeModel(wordmodel);  OneWordModel wordmodelInsertDB=SetupModelToInsertDB(mean, pronunciation, word, wordtype, searchOptions);  VocabularyDAO vocabularyDAO =**new** VocabularyDAO();  vocabularyDAO.add(MyUtils.*getStoredConnection*(request), wordmodelInsertDB);  //Tạo một model word phù hợp với cây nhị phân việt anh  InserModelToAnotherBinary(mean, pronunciation, word, wordtype, treeModelVA);  MyUtils.*storeBinaryTreeVA*(request.getSession(), treeModelVA);  url="/views/web/home.jsp";  } |
| Giải thích code | Ta tiến hành kiểm tra xem là đang thực hiện thêm dữ liệu ở trang là Việt Anh hay Anh việt. Trong trường học từ thêm thuộc Anh Việt ta tiếng anh tạo một model phù hợp với cây nhị phân Anh Việt.Sau đó tiến anh thêm node chứa từ cần thêm vào bên trong cây nhị phân Anh Việt và tiếng hành cân bằng cây nhị phân ở đây là hàm insertAVL.  Trong trường hợp khác tức là cây nhị phân của từ mới thêm là cây nhị phân Việt Anh thì nhóm thực hiện tiến hành tạo model phù hợp và thêm vào cây tương tự như trên. |
| Code java | **public** **void** InsertModelToBinary(String mean,String pronunciation,String word,String wordtype,String searchOptions,BinaryTreeModel treeModel)  {  //Ở hàm này ta thêm cái node vào đúng binary tree mà ta đang làm việc trên đó  **if**(searchOptions.equals("1"))  {  OneWordModel wordmodelInsertDB= **new** OneWordModel();  wordmodelInsertDB.setWord(word);  wordmodelInsertDB.setMean(mean);  wordmodelInsertDB.setPronunciation(pronunciation);  wordmodelInsertDB.setWordType(wordtype);  BinaryTree binaryTree=**new** BinaryTree();  binaryTree.insertAVL(treeModel.getRoot(), wordmodelInsertDB);  }  **else**  {  OneWordModel wordmodel= **new** OneWordModel();  wordmodel.setWord(word);  wordmodel.setMean(mean);  wordmodel.setPronunciation(pronunciation);  wordmodel.setWordType(wordtype);  //Insert vào cây nhị phân Việt Anh  BinaryTree binaryTree=**new** BinaryTree();  binaryTree.insertAVL(treeModel.getRoot(), wordmodel);  }  } |
| Giải thích code | Ta tiến hành kiểm tra xem là đang thực hiện thêm dữ liệu ở trang là Việt Anh hay Anh việt. Trong trường học từ thêm thuộc Anh Việt(searchOptions=1) ta tiến hành tạo một model phù hợp với cây nhị phân Việt Anh .Sau đó tiến anh thêm node chứa từ cần thêm vào bên trong cây nhị phân Việt Anh và tiếng hành cân bằng cây nhị phân ở đây là hàm insertAVL. |
| Code java | **public** **void** InserModelToAnotherBinary(String mean,String pronunciation,String word,String wordtype,BinaryTreeModel treeModel)  {  //Ở hàm này ta làm ngược là addmodel vừa tìm được vào cây còn lại (update cây nhị phân còn lại)  //Cây nhị phân đưa vào sẽ là khác với cây nhị phân với search options  //Ta có 1 sẽ là số kí hiệu cây nhị phân Anh Việt thì treemodel sẽ là cây nhị phân Việt Anh  BinaryTree binaryTree=**new** BinaryTree();  OneWordModel wordmodel= **new** OneWordModel();  wordmodel.setWord(mean);  wordmodel.setMean(word);  wordmodel.setPronunciation(pronunciation);  wordmodel.setWordType(wordtype);  binaryTree.insertAVL(treeModel.getRoot(), wordmodel);  } |
| Giải thích code | Nhóm thực hiện tiến hành tạo một model phù hợp với database đã đề ra. Việc chọn model phù hợp với database dựa vào từ đang thêm vào thuộc cây nhị phân nào(Anh Việt searchOptions=1 và Việt Anh searchOptions=2) sao đó trả về giá trị model phù hợp với database. |
| Code java | **public** OneWordModel SetupModelToInsertDB(String mean,String pronunciation,String word,String wordtype,String searchOptions)  {  OneWordModel wordmodelInsertDB= **new** OneWordModel();  **if**(searchOptions.equals("1"))  {  wordmodelInsertDB.setWord(word);  wordmodelInsertDB.setMean(mean);  wordmodelInsertDB.setPronunciation(pronunciation);  wordmodelInsertDB.setWordType(wordtype);  }  **else**  {  wordmodelInsertDB.setWord(mean);  wordmodelInsertDB.setMean(word);  wordmodelInsertDB.setPronunciation(pronunciation);  wordmodelInsertDB.setWordType(wordtype);  }  **return** wordmodelInsertDB;  } |
| Giải thích code | Hàm được dùng để kiểm tra các giá trị đầu vào. Ở đây cách kiểm tra các input tương tự nên nhóm thưc hiện chỉ demo một số code trong bài báo cáo. Đầu tiên phát hiện xem từ nhập vào có giá trị rỗng hoặc null hay không. Tiếp theo là thực hiện việc kiểm tra xem từ vừa đươc thêm đã có trong database phù hợp hay chưa. Cuối cùng sẽ trả về giá trị là đã xảy ra lỗi hay chưa. |
| Code trên java | **public** **boolean** CheckErrorInput(String mean,String pronunciation,String word,String wordtype,CheckError checkError,BinaryTreeModel treeModel)  {  **boolean** error=**false**;  **if**(!CheckError.*CheckMean*(mean))  {  checkError.setErrorMean("Please enter mean");  error=**true**;  }  **else**  {  **if**(!CheckError.*HaveNumber*(mean))  {  error=**true**;  checkError.setErrorMean("You must enter word not enter number");  }  }  **if**(CheckError.*CheckWord*(word))  {  BinaryTree binaryTree=**new** BinaryTree();  NodeModel nodeModel=binaryTree.Search(treeModel.getRoot(), word);  **if**(nodeModel!=**null**)  {  error=**true**;  checkError.setHaveNode("Word already exist");  }  }  **return** error;  } |

* EditWordServlet.java

|  |  |
| --- | --- |
| Giải thích code | Đầu tiên nhóm thực hiện thực hiện các thao tác như việc cho phép nhận các dữ liệu có dấu. Đồng thời tiến hành lấy dữ liệu của action(hành động thực hiện) đồng thời lấy dữ liệu của cây nhị phân Anh Việt(treeModel) và cây nhị phân Việt Anh(treeModelVA), đồng thời lấy biến searchOptions để xác định xem thao tác được thực hiện trên cây nhị phân nào.  Tiếp theo là các thao tác khi action nhận vào các hành động. |
| Code trên java | request.setCharacterEncoding(“UTF-8”);  String action=request.getParameter(“action”);  String searchOptions = request.getParameter(“searchOptions”);  BinaryTreeModel treeModel=MyUtils.*getBinaryTree*(request.getSession());  BinaryTreeModel treeModelVA=MyUtils.*getBinaryTreeForViet\_Anh*(request.getSession());  String url=””; |
| Giải thích code | Khi action nhận vào hành động là update. Tức có một từ được người dùng yêu cầu hiển thị thông tin để điều chỉnh update. Với searchOptions=1 tức là dữ liệu đang làm việc trên cây nhị phân Anh Việt thì ta tiến hành lấy dữ liệu của từ cần chỉnh sửa. Tìm nodeModel của từ đó trong cây nhị phân và tiến hành lưu trữ dữ liệu để trang hiển thị thuận lợi để lấy dữ liệu, cuối cùng là gán url bằng trang hiển thị tương ứng của cây nhị phân đó.  Trong trường hợp searchOptions=2 tức dữ liệu đang được thực hiện trên cây nhị phân Việt Anh thì ta tiến hành làm tương tự như trên. |
| Code trên java | **else** **if**(action.equals("update"))  {  **if**(searchOptions.equals("1"))  {  String word=request.getParameter("WordEdit").trim();  BinaryTree binaryTree=**new** BinaryTree();  NodeModel nodeModel=binaryTree.Search(treeModel.getRoot(), word);  request.setAttribute("nodeModel", nodeModel);  url="/views/web/EditWord.jsp";  }  **else**  {  //Cây Việt ANh chưa thực hiện update.  String word=request.getParameter("WordEdit").trim();  BinaryTree binaryTree=**new** BinaryTree();  NodeModel nodeModel=binaryTree.Search(treeModelVA.getRoot(), word);  request.setAttribute("nodeModel", nodeModel);  url="/views/web/EditWord2.jsp";  System.***out***.println("Go to Edit Word V-A");  }  } |
| Giải thích code | Khi action được đưa vào hoạt động remove thì servlet sẽ tiến hành thực hiện. Đầu tiên kiểm tra xem việc remove trên đang hoạt động ở cây nào trong trường hợp hoạt động ở cây nhị phân Anh Việt. Nhóm thực hiện tiếng anh việc lấy dữ liệu được trả về là word (từ được chỉ định để xoá) sau đó trong cây nhị phân tiến hành tìm kiếm node trong cây nhị phân đó thông qua hàm search truyền cây nhị phân Anh Việt và từ cần tìm vào. Sau đó web tiến hành việc xoá dữ liệu ra khỏi cây nhị phân Anh Việt.  Tiếp theo ta tiến hành xoá dữ liệu của từ chứa trong node vừa tìm được thông qua hàm delete trong class vocabularyDAO. Lưu trữ cây nhị phân Anh Việt vừa xoá.  Bước kế tiếp, chương trình tiến hành tìm xoá node chứa từ vừa được chỉ định xoá trong cây nhị phân Việt Anh còn lại. Thực hiện bằng hàm delete và tiến hành đưa cây nhị phân Việt Anh và từ cần xoá trong cây nhị phân đó (theo cấu trúc thì từ cần xoá trong cây nhị phân Việt Anh sẽ tương ứng với nghĩa của từ trong cây nhị phân Anh Việt). Cuối cùng lưu trữ cây nhị phân Việt Anh. Đồng thời chuyên hướng đến trang search của cây nhị phân tương ứng.  Đối với searchOptions=2 tức đang thực hiện trên cây nhị phân Việt Anh thì chương trình thực hiện các thao tác tương tự đã nêu ở trên. |
| Code trên java | **else** **if**(action.equals("Remove"))  {  **if**(searchOptions.equals("1"))  {  String word=request.getParameter("WordEdit").trim();  BinaryTree binaryTree=**new** BinaryTree();  NodeModel nodeModel=binaryTree.Search(treeModel.getRoot(), word);  NodeModel model=binaryTree.deleteNode(treeModel.getRoot(), word);  VocabularyDAO vocabularyDAO=**new** VocabularyDAO();  vocabularyDAO.delete(MyUtils.*getStoredConnection*(request), nodeModel.getData().getWord());  MyUtils.*storeBinaryTree*(request.getSession(), treeModel);//lưu lại cái cây  NodeModel modelVA=binaryTree.deleteNode(treeModelVA.getRoot(), nodeModel.getData().getMean());  MyUtils.*storeBinaryTreeVA*(request.getSession(), treeModelVA);  url="/views/web/home.jsp";  }  **else**  {  String word=request.getParameter("WordEdit").trim();  BinaryTree binaryTree=**new** BinaryTree();  NodeModel nodeModel=binaryTree.Search(treeModelVA.getRoot(), word);  List<String> list=binaryTree.find\_NLR(treeModelVA.getRoot(), word);  NodeModel model=binaryTree.deleteNode(treeModelVA.getRoot(), word);  VocabularyDAO vocabularyDAO=**new** VocabularyDAO();  vocabularyDAO.delete(MyUtils.*getStoredConnection*(request), nodeModel.getData().getMean());  MyUtils.*storeBinaryTreeVA*(request.getSession(), treeModelVA);//lưu lại cái cây  NodeModel modelVA=binaryTree.deleteNode(treeModel.getRoot(), nodeModel.getData().getMean());  MyUtils.*storeBinaryTree*(request.getSession(), treeModel);  url="/views/web/home2.jsp";  } |
| Giải thích code | Trong trường hợp hành động được đưa vào là edit(chỉnh sửa từ) với các giá trị phù hợp. Chương trình tiến hanh kiểm tra xem chương trình đang làm việc với cây nhị phân nào thông qua tham số searchOptions.  Trường hợp searchOptions bằng 1 thì chương trình đang làm với cây nhị phân Anh Việt ta tiến hành các dữ liệu đã được truyền qua như word(từ được điều chỉnh và không thay đổi giá trị),wordtype,pronunciation,mean. Với các giá trị trên chương trình tiến hành tạo tìm kiếm node chứa từ cần chỉnh sửa bên trong cây nhị phân tương ứng.  Sau khi tìm kiếm được thì chương trình tiến hành điều chỉnh lại từ chứa trong node của cây thông qua các hàm đã được nghĩa ở trên.  Sau đó tiến hành điều chỉnh từ đó bên trong databse bằng hàm edit và đưa cá giá trị như Word model tương ứng với node và từ mà ta cần chỉnh sửa.  Tiếp theo ta tiến hành cập nhật trên cây nhị phân còn lại(trong trường hợp optonsSearch=1 thì cây nhị phân còn lại là cây nhị phân Việt Anh).Đầu tiên ta tiến hành tìm kiếm node trên cây nhị phân Việt Anh thông qua tham số WordMean(tương ứng với word bên cây nhị phân Việt Anh), sau khi tìm kiếm được ta tiến hành thay đổi các giá trị phù hợp với các giá trị chỉnh sửa tương tự như trên. Cuối cùng chương trình sẽ tiến hành lưu trữ lại cây nhị phân Việt Anh.  Trong trường hợp ta có searchOptions bằng 2 tức chương trình đang thực hiện thao tác trên cây nhị phân Việt Anh thì chương trình sẽ được thực hiện tượng tự các hoạt động như đã trình bày ở trên. |
| Code trên java | **if**(action.equals("Edit"))  {  **if**((searchOptions.equals("1")))  {  BinaryTree binaryTree=**new** BinaryTree();  String word=request.getParameter("wordSearch").trim();  String pronunciation=request.getParameter("Pronunciation").trim();  String wordtype=request.getParameter("WordType").trim();  String mean=request.getParameter("Mean").trim();  NodeModel nodeModel=binaryTree.Search(treeModel.getRoot(), word);  nodeModel.getData().setWord(word);  nodeModel.getData().setMean(mean);  nodeModel.getData().setPronunciation(pronunciation);  nodeModel.getData().setWordType(wordtype); VocabularyDAO vocabularyDAO =**new** VocabularyDAO();  **boolean** rs=vocabularyDAO.edit(MyUtils.*getStoredConnection*(request), nodeModel.getData(),nodeModel.getData().getWord());  MyUtils.*storeBinaryTree*(request.getSession(), treeModel);  String wordmean=request.getParameter("wordMean").trim();  NodeModel nodeModelVA=binaryTree.Search(treeModelVA.getRoot(), wordmean);  nodeModelVA.getData().setWord(mean);  nodeModelVA.getData().setMean(word);  nodeModelVA.getData().setPronunciation(pronunciation);  nodeModelVA.getData().setWordType(wordtype);  MyUtils.*storeBinaryTreeVA*(request.getSession(), treeModelVA);  url="/views/web/home.jsp";  }  **else**  {  System.***out***.println("Viet-Anh edit");  BinaryTree binaryTree=**new** BinaryTree();  String word=request.getParameter("wordSearch").trim();  String pronunciation=request.getParameter("Pronunciation").trim();  String wordtype=request.getParameter("WordType").trim();  String mean=request.getParameter("Mean").trim();  NodeModel nodeModelVA=binaryTree.Search(treeModelVA.getRoot(), word);  nodeModelVA.getData().setWord(word);  nodeModelVA.getData().setMean(mean);  nodeModelVA.getData().setPronunciation(pronunciation);  nodeModelVA.getData().setWordType(wordtype);  VocabularyDAO vocabularyDAO =**new** VocabularyDAO();  MyUtils.*storeBinaryTreeVA*(request.getSession(), treeModelVA);  String wordmean=request.getParameter("wordMean").trim();  NodeModel nodeModel=binaryTree.Search(treeModel.getRoot(), wordmean);  nodeModel.getData().setWord(mean);  nodeModel.getData().setMean(word);  nodeModel.getData().setPronunciation(pronunciation);  nodeModel.getData().setWordType(wordtype);  **boolean** rs=vocabularyDAO.edit(MyUtils.*getStoredConnection*(request), nodeModel.getData(),word);  MyUtils.*storeBinaryTree*(request.getSession(), treeModel);  url="/views/web/home2.jsp";  } |

## VI. Tài liệu tham khảo

[1]. <https://wiki.matbao.net/apache-la-gi-tomcat-la-gi-huong-dan-cai-dat-chi-tiet-nhat/#apache-la-gi>

[2]. AVL Tree set 1, GreeksforGreek, https://www.geeksforgeeks.org/avl-tree-set-1-insertion/