**TRƯỜNG ĐẠI HỌC LÂM NGHIỆP**

KHOA KINH TẾ VÀ QUẢN TRỊ KINH DOANH

**===o0o===**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: LẬP TRÌNH MẠNG**

**ĐỀ TÀI:**

**PHÁT TRIỂN PHẦN MỀM DỊCH THUẬT TỪ API GOOGLE TRANSLATE**



**Giảng viên hướng dẫn: Mai Hà An**

**Sinh viên thực hiện:**

**Họ và Tên: Nguyễn Khả Đăng**

**Mã sinh viên: 2041070501**

**Lớp: K65 - HTTT**

**Hà Nội, 2022**

MỤC LỤC

[CHƯƠNG I. TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU 4](#_Toc133584614)

[1.1 Lý do, mục đích chọn đề tài 4](#_Toc133584615)

[1.2 Mục tiêu và phạm vi nghiên cứu 4](#_Toc133584616)

[1.2.1 Mục tiêu 4](#_Toc133584617)

[1.2.2 Phạm vi nghiên cứu 5](#_Toc133584618)

[CHƯƠNG II. CƠ SỞ LÝ THUYẾT 6](#_Toc133584619)

[2.1. Lập trình mạng 6](#_Toc133584620)

[2.1.1 Định Nghĩa 6](#_Toc133584621)

[2.1.2 Những ngôn ngữ lập trình mạng 6](#_Toc133584622)

[2.1.3 Các mô hình lập trình mạng 7](#_Toc133584623)

[2.2 API 7](#_Toc133584624)

[2.2.1 Định Nghĩa 7](#_Toc133584625)

[2.2.2 Một số loại API phổ biến 7](#_Toc133584626)

[CHƯƠNG III. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 8](#_Toc133584627)

[3.1 Thiết kế mô hình UML 8](#_Toc133584628)

[3.1.1 Danh sách các đối tượng sử dụng (Actor) 8](#_Toc133584629)

[3.1.2 Mô tả các use case 8](#_Toc133584630)

[3.1.3 Cơ sở dữ liệu 10](#_Toc133584631)

[3.1.4. Biểu đồ phân rã các UC 10](#_Toc133584632)

[3.1.5. Biểu đồ hoạt động 11](#_Toc133584633)

[CHƯƠNG IV. LẬP TRÌNH VÀ KẾT QUẢ 14](#_Toc133584634)

[4.1. Các hàm chức năng 14](#_Toc133584635)

[4.1.1 Các hàm tương tác với cơ sở dữ liệu 14](#_Toc133584636)

[4.1.2 Hàm chuyển đổi tên ngôn ngữ thành mã ngôn ngữ 14](#_Toc133584637)

[4.1.3 Hàm chuyển đổi mã thành tên ngôn ngữ 16](#_Toc133584638)

[4.1.4 Hàm tạo request tới API Google Translate 18](#_Toc133584639)

[4.1.5 Hàm tạo request phát hiện ngôn ngữ 19](#_Toc133584640)

[4.1.6 ORC 20](#_Toc133584641)

[4.1.7 Hàm tạo request API zalo.ai để phát âm thanh 22](#_Toc133584642)

[4.1.8 Hàm nhập bằng voice 22](#_Toc133584643)

[4.2. Kết quả lập trình 24](#_Toc133584644)

[4.2.1 Main Form 24](#_Toc133584645)

[4.2.2 Properties tab 24](#_Toc133584646)

[CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 25](#_Toc133584647)

[5.1. Kết luận 25](#_Toc133584648)

[5.1.1. Kết quả đạt được 25](#_Toc133584649)

[5.1.2 Hạn Chế 25](#_Toc133584650)

[5.2. Hướng phát triển 25](#_Toc133584651)

[TÀI LIỆU THAM KHẢO 26](#_Toc133584652)

# CHƯƠNG I. TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU

## 

## 1.1 Lý do, mục đích chọn đề tài

1. Nhu cầu dịch thuật của người dùng: Hiện nay, trong thế giới liên kết toàn cầu, việc giao tiếp và trao đổi thông tin trở nên ngày càng quan trọng. Tuy nhiên, không phải ai cũng biết ngoại ngữ hoặc có thể nói, viết và đọc được các ngôn ngữ khác nhau. Do đó, phần mềm dịch thuật trên desktop sẽ giúp cho người dùng dễ dàng truy cập và sử dụng các tài liệu, thông tin, văn bản, email, tin nhắn, v.v. bằng cách dịch nhanh và chính xác các ngôn ngữ khác nhau.
2. Sự tiện lợi và nhanh chóng: Phần mềm dịch thuật trên desktop sẽ giúp cho người dùng tiết kiệm thời gian và công sức khi cần dịch các văn bản, thông tin hay trao đổi với người nước ngoài. Thay vì phải tìm kiếm các công cụ dịch thuật trực tuyến trên trình duyệt, người dùng có thể sử dụng phần mềm dịch thuật trên desktop với giao diện thân thiện và dễ sử dụng hơn.
3. API Google Translate đáp ứng được nhu cầu dịch thuật của nhiều ngôn ngữ: API Google Translate hỗ trợ hơn 100 ngôn ngữ, cho phép người dùng dịch thuật giữa các ngôn ngữ khác nhau. Điều này rất hữu ích cho những người làm việc trong môi trường đa ngôn ngữ hoặc cần phải liên lạc với đối tác, khách hàng từ các quốc gia khác nhau.
4. Đáp ứng yêu cầu của đồ án môn lập trình mạng: Việc phát triển phần mềm dịch thuật trên desktop sử dụng API Google Translate sẽ giúp sinh viên hiểu và áp dụng các kiến thức về lập trình mạng vào thực tế. Đồng thời, đây cũng là cơ hội để sinh viên rèn luyện kỹ năng lập trình, phát triển phần mềm và tương tác với API.

## 1.2 Mục tiêu và phạm vi nghiên cứu

### 1.2.1 Mục tiêu

Phát triển một phần mềm dịch thuật trên desktop bằng API Google Translate, với khả năng dịch thuật giữa các ngôn ngữ khác nhau và đáp ứng nhu cầu sử dụng của người dùng.

### 1.2.2 Phạm vi nghiên cứu

Phạm vi nghiên cứu của bài là xây dựng một phần mềm dịch thuật trên desktop sử dụng API Google Translate để dịch thuật giữa các ngôn ngữ khác nhau. Phần mềm này sẽ được thiết kế với giao diện đơn giản và dễ sử dụng, cho phép người dùng dễ dàng nhập văn bản cần dịch và chọn ngôn ngữ đích và nguồn. Sau đó, phần mềm sẽ sử dụng API Google Translate để dịch thuật và hiển thị kết quả dịch thuật cho người dùng.

Nghiên cứu này sẽ tập trung vào phát triển phần mềm dịch thuật trên desktop, vì đây là một phương tiện tiện lợi và phổ biến để người dùng có thể sử dụng trong các hoạt động giao tiếp, trao đổi thông tin và làm việc trong môi trường đa ngôn ngữ. Tuy nhiên, nghiên cứu này sẽ không bao gồm phát triển phần mềm trên các nền tảng di động hoặc các ứng dụng web.

# CHƯƠNG II. CƠ SỞ LÝ THUYẾT

## Lập trình mạng

### 2.1.1 Định Nghĩa

Lập trình mạng là nhiệm vụ cơ bản để phát triển các ứng dụng trong hệ thống doanh nghiệp, từ chương trình phần mềm quản lý như kế toán, nhân sự,… cho đến ứng dụng giải trí là trò chơi, điều khiển…

Lập trình mạng được xây dựng dựa trên công thức:

*Lập trình mạng (LTM) = Kiến thức mạng truyền thông + Mô hình LTM + Ngôn ngữ LTM*

Dựa theo công thức trên, ta có thể thấy có ba vấn đề chính cốt lõi là kiến thức mạng truyền thông, mô hình lập trình mạng và ngôn ngữ lập trình mạng.

Kiến thức mạng truyền thông là những kiến thức về mạng điện thoại di động, PSTN, hệ thống GPS, mạng như BlueTooth, WUSB, mạng sensor…. Nhất là cách sử dụng cũng như khai thác chúng.

Mô hình lập trình mạng là những kiến thức về cách xây dựng hệ thống mạng, mô hình xây dựng chương trình ứng dụng mạng, kiến thức về cơ sở dữ liệu…

Cuối cùng là ngôn ngữ lập trình mạng, đây là yếu tố quyết định xem các chương trình mạng của bạn sẽ hoạt động như thế nào. Có rất nhiều ngôn ngữ lập trình mạng khác nhau, tùy theo mục đích sử dụng.

### 2.1.2 Những ngôn ngữ lập trình mạng

Hiện nay, có nhiều loại ngôn ngữ lập trình khác nhau để xây dựng một ứng dụng mạng. Chúng đều tồn tại các ưu - nhược điểm khác nhau. Nhưng chung quy vẫn là hỗ trợ thư viện API với nhiều cấp độ và mức phụ thuộc cụ thể.

Một số ngữ lập trình mạng phổ biến và được sử dụng nhiều nhất hiện nay là Java, .NET, C/C++, Delphi và JavaScript.

Trước đây, Java và .NET là hai ngôn ngữ lập trình được sử dụng rộng rãi. Nhưng cho đến ngày nay, đã xuất hiện thêm nhiều ngôn ngữ với vô số tiện ích. Đặc biệt, có thể xây dựng ứng dụng mạng một cách nhanh chóng và dễ dàng.

Tiêu biểu nhất là Javascript, một làn sóng xu hướng mới cho lập trình Full Stack. Nó đã và đang dần thay thế cho công nghệ Front End hay Back End truyền thống.

### 2.1.3 Các mô hình lập trình mạng

Mô hình Client-Server: Đây là mô hình lập trình mạng cơ bản nhất, trong đó có hai thành phần chính là client (người dùng) và server (máy chủ). Client gửi yêu cầu đến server và server phản hồi bằng cách cung cấp thông tin hoặc dịch vụ. Ví dụ: trang web, ứng dụng email, game online...

Mô hình Peer-to-Peer (P2P): Trong mô hình này, không có server trung tâm và các thiết bị sẽ liên kết với nhau để chia sẻ tài nguyên và thông tin. Mỗi thiết bị trong mạng có thể hoạt động như là một client hoặc một server, tùy thuộc vào hoạt động hiện tại của nó. Ví dụ: BitTorrent, Skype...

Mô hình MVC (Model-View-Controller): Đây là một mô hình thiết kế phần mềm, được sử dụng trong lập trình web. Mô hình này chia thành ba phần: Model (mô hình) đại diện cho dữ liệu, View (giao diện) đại diện cho tầng giao diện người dùng và Controller (bộ điều khiển) quản lý việc tương tác giữa Model và View. Ví dụ: Ruby on Rails, Django...

Mô hình REST (Representational State Transfer): Đây là một kiến trúc được sử dụng trong các ứng dụng web để tương tác với các dịch vụ web. REST tập trung vào tài nguyên được đại diện bởi các địa chỉ URL và các phương thức HTTP được sử dụng để tương tác với các tài nguyên đó. Ví dụ: Twitter, Facebook API...

## 2.2 API

### 2.2.1 Định Nghĩa

API (Application Programming Interface) là một tập hợp các quy tắc, giao thức và công cụ được sử dụng để phát triển các ứng dụng phần mềm và cho phép các ứng dụng khác tương tác với ứng dụng của bạn. API thường được sử dụng để chia sẻ dữ liệu và chức năng giữa các ứng dụng khác nhau và cung cấp tính năng mở rộng cho ứng dụng của bạn.

### 2.2.2 Một số loại API phổ biến

* API Web: API Web là một loại API sử dụng giao thức HTTP để cho phép các ứng dụng tương tác với các dịch vụ web như Google Maps, Facebook, Twitter, và PayPal.
* API RESTful: REST (Representational State Transfer) là một kiến trúc được sử dụng để phát triển các dịch vụ web. API RESTful là một loại API sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các yêu cầu và truy xuất dữ liệu.
* API SOAP: SOAP (Simple Object Access Protocol) là một giao thức được sử dụng để truyền thông tin giữa các ứng dụng. API SOAP sử dụng XML để định dạng dữ liệu và cung cấp các tính năng bảo mật và kiểm soát phiên.
* API nhúng: API nhúng được sử dụng để kết nối các thiết bị nhúng như cảm biến, thiết bị điều khiển và robot với các ứng dụng.

# CHƯƠNG III. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 3.1 Thiết kế mô hình UML

### 3.1.1 Danh sách các đối tượng sử dụng (Actor)

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Mô tả |
| 1 | User | Là người sử dụng phần mềm dịch thuật |
| 2 | Google Translate API | Là API cung cấp dịch vụ dịch thuật của Google được sử dụng trong phần mềm |
| 3 | Hệ thống nhận dạng giọng nói | Là hệ thống nhận dạng giọng nói của hệ điều hành được sử dụng để nhập văn bản bằng giọng nói |
| 4 | Hệ thống OCR | Là hệ thống nhận dạng văn bản từ ảnh được sử dụng để lấy văn bản từ ảnh trong tính năng OCR |
| 5 | Zalo.ai API | Là API cung cấp dịch vụ đọc văn bản của Zalo.ai được sử dụng trong phần mềm |

Bảng 1: Danh sách các đối tượng sử dụng (Actor)

### 3.1.2 Mô tả các use case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case | Tên use case | Mô tả | Pre-condition | Post-condition |
| UC-01 | Dịch văn bản | Người dùng nhập văn bản cần dịch và chọn ngôn ngữ dịch | Phần mềm đã được khởi động | Văn bản đã được dịch và hiển thị kết quả cho người dùng |
| UC-02 | Phát hiện ngôn ngữ | Phần mềm phát hiện ngôn ngữ của văn bản đầu vào | Phần mềm đã được khởi động | Ngôn ngữ của văn bản được hiển thị cho người dùng |
| UC-03 | Nhập bằng voice | Người dùng sử dụng giọng nói để nhập văn bản cần dịch | Phần mềm đã được khởi động và đã được cấp quyền truy cập microphone | Văn bản đã được dịch và hiển thị kết quả cho người dùng |
| UC-04 | Đọc văn bản | Phần mềm đọc văn bản đầu vào cho người dùng | Phần mềm đã được khởi động | Văn bản đã được đọc cho người dùng |
| UC-05 | OCR chụp và lấy văn bản từ ảnh | Người dùng sử dụng tính năng OCR để chụp ảnh văn bản và lấy văn bản đó | Phần mềm đã được khởi động và đã được cấp quyền truy cập máy ảnh | Văn bản đã được lấy và hiển thị kết quả cho người dùng |

Bảng 2: Danh sách các use case chính

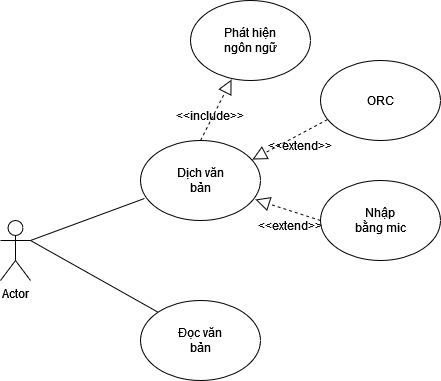
### 3.1.3 Cơ sở dữ liệu

Bảng Keys

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Mô tả |
| idEvent | nchar | Khoá chính |
| keys | nchar | Tên đăng nhập |

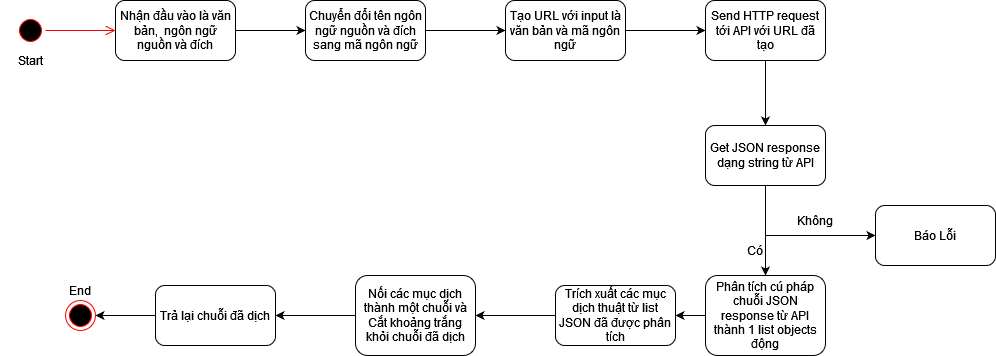
### 3.1.4. Biểu đồ phân rã các UC

* **Biểu đồ Usecase tổng quát**

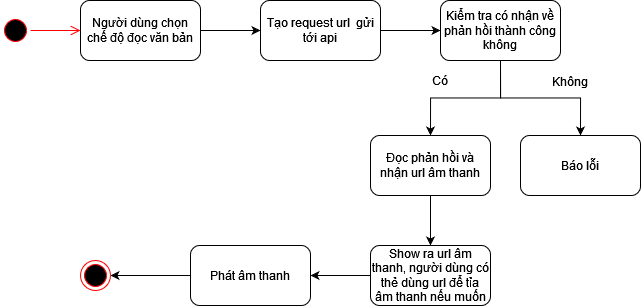
****

### 3.1.5. Biểu đồ hoạt động

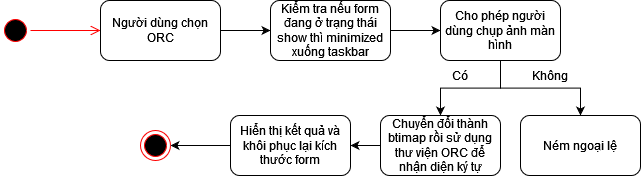
* **Biểu đồ hoạt động dịch văn bản**

****

* **Biểu đồ hoạt động đọc văn bản**

****

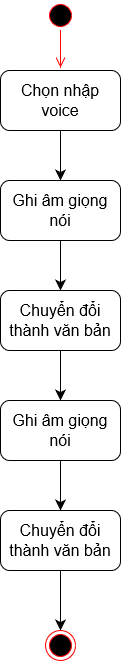
* **Biểu đồ hoạt đông ORC**

****

* **Biểu đồ hoạt động nhận dạng ngôn ngữ**

****

* **Biểu đồ hoạt động nhập bằng voice**

****

# CHƯƠNG IV. LẬP TRÌNH VÀ KẾT QUẢ

## 4.1. Các hàm chức năng

### 4.1.1 Các hàm tương tác với cơ sở dữ liệu

public static SQLiteConnection GetDBConnection()

{

var sql = "Data Source = KeyPress.db";

SQLiteConnection con = new SQLiteConnection(sql);

return con;

}

SQLiteConnection con = GetDBConnection();

public void CreateDB()

{

var sql = "Data Source = KeyPress.db";

SQLiteConnection.CreateFile(sql);

}

public void CreateTable(SQLiteConnection con)

{

string create = "CREATE TABLE Keys(idEvent NCHAR(10) PRIMARY KEY, keys NCHAR(10))";

var cmd = new SQLiteCommand(create, con);

cmd.ExecuteNonQuery();

}

public void Insert(SQLiteConnection con, string idEvent, string keys)

{

string insert = "INSERT INTO Keys(idEvent, keys) VALUES(@idEvent, @keys)";

var cmd = new SQLiteCommand(insert, con);

cmd.Parameters.AddWithValue("@idEvent", idEvent);

cmd.Parameters.AddWithValue("@keys", keys);

cmd.ExecuteNonQuery();

}

public void Update(string idEvent, string keys)

{

if (con.State == System.Data.ConnectionState.Closed)

{

con.Open();

}

string update = "UPDATE Keys SET keys = @keys WHERE idEvent = @idEvent";

SQLiteCommand cmd = new SQLiteCommand(update, con);

cmd.Parameters.AddWithValue("@keys", keys);

cmd.Parameters.AddWithValue("@idEvent", idEvent);

cmd.ExecuteNonQuery();

con.Close();

}

}

### 4.1.2 Hàm chuyển đổi tên ngôn ngữ thành mã ngôn ngữ

public string ConvertLanguageNameToCode(string languageName)

{

string code = "";

//110 ngôn ngữ

switch (languageName)

{

case "Afrikaans": code = "af"; break;

case "Albanian": code = "sq"; break;

case "Amharic": code = "am"; break;

case "Arabic": code = "ar"; break;

case "Armenian": code = "hy"; break;

case "Azerbaijani": code = "az"; break;

case "Basque": code = "eu"; break;

case "Belarusian": code = "be"; break;

case "Bengali": code = "bn"; break;

case "Bosnian": code = "bs"; break;

case "Bulgarian": code = "bg"; break;

case "Catalan": code = "ca"; break;

case "Cebuano": code = "ceb"; break;

case "Chichewa": code = "ny"; break;

case "Chinese (Simplified)": code = "zh-CN"; break;

case "Chinese (Traditional)": code = "zh-TW"; break;

case "Corsican": code = "co"; break;

case "Croatian": code = "hr"; break;

case "Czech": code = "cs"; break;

case "Danish": code = "da"; break;

case "Dutch": code = "nl"; break;

case "English": code = "en"; break;

case "Esperanto": code = "eo"; break;

case "Estonian": code = "et"; break;

case "Filipino": code = "tl"; break;

case "Finnish": code = "fi"; break;

case "French": code = "fr"; break;

case "Frisian": code = "fy"; break;

case "Galician": code = "gl"; break;

case "Georgian": code = "ka"; break;

case "German": code = "de"; break;

case "Greek": code = "el"; break;

case "Gujarati": code = "gu"; break;

case "Haitian Creole": code = "ht"; break;

case "Hausa": code = "ha"; break;

case "Hawaiian": code = "haw"; break;

case "Hebrew": code = "iw"; break;

case "Hindi": code = "hi"; break;

case "Hmong": code = "hmn"; break;

case "Hungarian": code = "hu"; break;

case "Icelandic": code = "is"; break;

case "Igbo": code = "ig"; break;

case "Indonesian": code = "id"; break;

case "Irish": code = "ga"; break;

case "Italian": code = "it"; break;

case "Japanese": code = "ja"; break;

case "Javanese": code = "jw"; break;

case "Kannada": code = "kn"; break;

case "Kazakh": code = "kk"; break;

case "Khmer": code = "km"; break;

case "Kinyarwanda": code = "rw"; break;

case "Korean": code = "ko"; break;

case "Kurdish (Kurmanji)": code = "ku"; break;

case "Kyrgyz": code = "ky"; break;

case "Lao": code = "lo"; break;

case "Latin": code = "la"; break;

case "Latvian": code = "lv"; break;

case "Lithuanian": code = "lt"; break;

case "Luxembourgish": code = "lb"; break;

case "Macedonian": code = "mk"; break;

case "Malagasy": code = "mg"; break;

case "Malay": code = "ms"; break;

case "Malayalam": code = "ml"; break;

case "Maltese": code = "mt"; break;

case "Maori": code = "mi"; break;

case "Marathi": code = "mr"; break;

case "Mongolian": code = "mn"; break;

case "Myanmar (Burmese)": code = "my"; break;

case "Nepali": code = "ne"; break;

case "Norwegian": code = "no"; break;

case "Nyanja (Chichewa)": code = "ny"; break;

case "Odia (Oriya)": code = "or"; break;

case "Pashto": code = "ps"; break;

case "Persian": code = "fa"; break;

case "Polish": code = "pl"; break;

case "Portuguese": code = "pt"; break;

case "Punjabi": code = "pa"; break;

case "Romanian": code = "ro"; break;

case "Russian": code = "ru"; break;

case "Samoan": code = "sm"; break;

case "Scots Gaelic": code = "gd"; break;

case "Serbian": code = "sr"; break;

case "Sesotho": code = "st"; break;

case "Shona": code = "sn"; break;

case "Sindhi": code = "sd"; break;

case "Sinhala (Sinhalese)": code = "si"; break;

case "Slovak": code = "sk"; break;

case "Slovenian": code = "sl"; break;

case "Somali": code = "so"; break;

case "Spanish": code = "es"; break;

case "Sundanese": code = "su"; break;

case "Swahili": code = "sw"; break;

case "Swedish": code = "sv"; break;

case "Tajik": code = "tg"; break;

case "Tamil": code = "ta"; break;

case "Tatar": code = "tt"; break;

case "Telugu": code = "te"; break;

case "Thai": code = "th"; break;

case "Turkish": code = "tr"; break;

case "Turkmen": code = "tk"; break;

case "Ukrainian": code = "uk"; break;

case "Urdu": code = "ur"; break;

case "Uyghur": code = "ug"; break;

case "Uzbek": code = "uz"; break;

case "Vietnamese": code = "vi"; break;

case "Welsh": code = "cy"; break;

case "Xhosa": code = "xh"; break;

case "Yiddish": code = "yi"; break;

case "Yoruba": code = "yo"; break;

case "Zulu": code = "zu"; break;

default: break;

}

return code;

}

### 4.1.3 Hàm chuyển đổi mã thành tên ngôn ngữ

public string ConvertCodeToLanguageName(string code)

{

string languageName = "";

//110 ngôn ngữ

switch (code)

{

case "af": languageName = "Afrikaans"; break;

case "sq": languageName = "Albanian"; break;

case "am": languageName = "Amharic"; break;

case "ar": languageName = "Arabic"; break;

case "hy": languageName = "Armenian"; break;

case "az": languageName = "Azerbaijani"; break;

case "eu": languageName = "Basque"; break;

case "be": languageName = "Belarusian"; break;

case "bn": languageName = "Bengali"; break;

case "bs": languageName = "Bosnian"; break;

case "bg": languageName = "Bulgarian"; break;

case "ca": languageName = "Catalan"; break;

case "ceb": languageName = "Cebuano"; break;

case "ny": languageName = "Chichewa"; break;

case "zh-CN": languageName = "Chinese (Simplified)"; break;

case "zh-TW": languageName = "Chinese (Traditional)"; break;

case "co": languageName = "Corsican"; break;

case "hr": languageName = "Croatian"; break;

case "cs": languageName = "Czech"; break;

case "da": languageName = "Danish"; break;

case "nl": languageName = "Dutch"; break;

case "en": languageName = "English"; break;

case "eo": languageName = "Esperanto"; break;

case "et": languageName = "Estonian"; break;

case "tl": languageName = "Filipino"; break;

case "fi": languageName = "Finnish"; break;

case "fr": languageName = "French"; break;

case "fy": languageName = "Frisian"; break;

case "gl": languageName = "Galician"; break;

case "ka": languageName = "Georgian"; break;

case "de": languageName = "German"; break;

case "el": languageName = "Greek"; break;

case "gu": languageName = "Gujarati"; break;

case "ht": languageName = "Haitian Creole"; break;

case "ha": languageName = "Hausa"; break;

case "haw": languageName = "Hawaiian"; break;

case "iw": languageName = "Hebrew"; break;

case "hi": languageName = "Hindi"; break;

case "hmn": languageName = "Hmong"; break;

case "hu": languageName = "Hungarian"; break;

case "is": languageName = "Icelandic"; break;

case "ig": languageName = "Igbo"; break;

case "id": languageName = "Indonesian"; break;

case "ga": languageName = "Irish"; break;

case "it": languageName = "Italian"; break;

case "ja": languageName = "Japanese"; break;

case "jw": languageName = "Javanese"; break;

case "kn": languageName = "Kannada"; break;

case "kk": languageName = "Kazakh"; break;

case "km": languageName = "Khmer"; break;

case "rw": languageName = "Kinyarwanda"; break;

case "ko": languageName = "Korean"; break;

case "ku": languageName = "Kurdish (Kurmanji)"; break;

case "ky": languageName = "Kyrgyz"; break;

case "lo": languageName = "Lao"; break;

case "la": languageName = "Latin"; break;

case "lv": languageName = "Latvian"; break;

case "lt": languageName = "Lithuanian"; break;

case "lb": languageName = "Luxembourgish"; break;

case "mk": languageName = "Macedonian"; break;

case "mg": languageName = "Malagasy"; break;

case "ms": languageName = "Malay"; break;

case "ml": languageName = "Malayalam"; break;

case "mt": languageName = "Maltese"; break;

case "mi": languageName = "Maori"; break;

case "mr": languageName = "Marathi"; break;

case "mn": languageName = "Mongolian"; break;

case "my": languageName = "Myanmar (Burmese)"; break;

case "ne": languageName = "Nepali"; break;

case "no": languageName = "Norwegian"; break;

case "ny (Chichewa)": languageName = "Nyanja"; break;

case "or": languageName = "Odia (Oriya)"; break;

case "ps": languageName = "Pashto"; break;

case "fa": languageName = "Persian"; break;

case "pl": languageName = "Polish"; break;

case "pt": languageName = "Portuguese"; break;

case "pa": languageName = "Punjabi"; break;

case "ro": languageName = "Romanian"; break;

case "ru": languageName = "Russian"; break;

case "sm": languageName = "Samoan"; break;

case "gd": languageName = "Scots Gaelic"; break;

case "sr": languageName = "Serbian"; break;

case "st": languageName = "Sesotho"; break;

case "sn": languageName = "Shona"; break;

case "sd": languageName = "Sindhi"; break;

case "si": languageName = "Sinhala (Sinhalese)"; break;

case "sk": languageName = "Slovak"; break;

case "sl": languageName = "Slovenian"; break;

case "so": languageName = "Somali"; break;

case "es": languageName = "Spanish"; break;

case "su": languageName = "Sundanese"; break;

case "sw": languageName = "Swahili"; break;

case "sv": languageName = "Swedish"; break;

case "tg": languageName = "Tajik"; break;

case "ta": languageName = "Tamil"; break;

case "tt": languageName = "Tatar"; break;

case "te": languageName = "Telugu"; break;

case "th": languageName = "Thai"; break;

case "tr": languageName = "Turkish"; break;

case "tk": languageName = "Turkmen"; break;

case "uk": languageName = "Ukrainian"; break;

case "ur": languageName = "Urdu"; break;

case "ug": languageName = "Uyghur"; break;

case "uz": languageName = "Uzbek"; break;

case "vi": languageName = "Vietnamese"; break;

case "cy": languageName = "Welsh"; break;

case "xh": languageName = "Xhosa"; break;

case "yi": languageName = "Yiddish"; break;

case "yo": languageName = "Yoruba"; break;

case "zu": languageName = "Zulu"; break;

default: break;

}

return languageName;

}

### 4.1.4 Hàm tạo request tới API Google Translate

public string TranslateText(string input, string sourceLanguage, string targetLanguage)

{

Convert\_language\_code cv = new Convert\_language\_code(); //khởi tạo cho hàm chuyển đổi mã ngôn ngữ

string url = String.Format("https://translate.googleapis.com/translate\_a/single?client=gtx&sl={0}&tl={1}&dt=t&q={2}", cv.ConvertLanguageNameToCode(sourceLanguage), cv.ConvertLanguageNameToCode(targetLanguage), Uri.EscapeUriString(input)); //URL của API Google Translate

HttpClient httpClient = new HttpClient(); //Tạo một đối tượng HttpClient để gửi yêu cầu HTTP đến API Google Translate.

string result = httpClient.GetStringAsync(url).Result; //Gửi yêu cầu HTTP đến API Google Translate bằng cách gọi phương thức GetStringAsync trên đối tượng httpClient, truyền vào địa chỉ URL của API và đợi kết quả trả về bằng cách gọi phương thức Result.

if (string.IsNullOrWhiteSpace(result))

{

throw new Exception("No response received from Google Translate API.");

}

var jsonData = new JavaScriptSerializer().Deserialize<List<dynamic>>(result);//Sử dụng lớp JavaScriptSerializer để phân tích cú pháp chuỗi JSON phản hồi từ API Google Translate và chuyển đổi nó thành một danh sách các đối tượng động (dynamic) của các dòng và các phần tử trong mỗi dòng

var translationItems = jsonData[0];//Lấy danh sách các dòng chứa phần dịch của câu đầu vào từ đối tượng JSON phân tích được.

string translation = "";

foreach (object item in translationItems)//Sử dụng một vòng lặp để ghép các phần tử trong danh sách các dòng phần dịch thành một chuỗi duy nhất.

{

IEnumerable translationLineObject = item as IEnumerable;

IEnumerator translationLineString = translationLineObject.GetEnumerator();

translationLineString.MoveNext();

translation += string.Format(" {0}", Convert.ToString(translationLineString.Current));

}

if (translation.Length > 1) //Kiểm tra xem chuỗi phần dịch có dài hơn 1 ký tự hay không, nếu có thì xóa ký tự đầu tiên (ký tự trống) và trả về phần dịch. Nếu không, trả về chuỗi phần dịch không chỉnh sửa.

{

translation = translation.Substring(1);

}

return translation;

}

### 4.1.5 Hàm tạo request phát hiện ngôn ngữ

public static class LanguageDetector

{

public static async Task<string> DetectLanguageAsync(string text, string targetLanguage)

{

Convert\_language\_code cv = new Convert\_language\_code();

using (var client = new HttpClient())

{

// Tạo biến url chứa URL của API Google Translate để nhận diện ngôn ngữ, sử dụng biến cv để chuyển đổi ngôn ngữ đích thành mã ngôn ngữ

var url = $"https://translate.googleapis.com/translate\_a/single?client=gtx&sl=auto&tl={cv.ConvertLanguageNameToCode(targetLanguage)}&dt=t&q={Uri.EscapeDataString(text)}";

// Gửi yêu cầu HTTP GET tới URL của API Google Translate bằng đối tượng HttpClient

var response = await client.GetAsync(url);

// Kiểm tra xem phản hồi từ API có thành công hay không, nếu không thì ném ra một ngoại lệ

response.EnsureSuccessStatusCode();

// Đọc nội dung của phản hồi HTTP từ API Google Translate và lưu trữ nó trong biến content dưới dạng chuỗi

var content = await response.Content.ReadAsStringAsync();

// Tách chuỗi JSON phản hồi từ API Google Translate để lấy kết quả phân tích ngôn ngữ

var parts = content.Split('"');

// Nếu chuỗi JSON phản hồi chứa ít nhất 2 phần tử, trả về phần tử thứ hai là kết quả phân tích ngôn ngữ

if (parts.Length >= 2)

{

return parts[1];

}

// Nếu không, ném ra một ngoại lệ

else

{

throw new Exception("Cannot detect language.");

}

}

}

}

### 4.1.6 ORC

public static Bitmap ConvertToBitmap(BitmapSource bitmapSource)

{

// Lấy chiều rộng và chiều cao của ảnh

var width = bitmapSource.PixelWidth;

var height = bitmapSource.PixelHeight;

// Tính toán kích thước mỗi hàng ảnh, theo đơn vị byte (stride)

var stride = width \* ((bitmapSource.Format.BitsPerPixel + 7) / 8);

// Cấp phát một vùng nhớ không quản lý (unmanaged memory) với kích thước bằng với kích thước ảnh.

// Biến memoryBlockPointer trỏ đến địa chỉ bắt đầu của vùng nhớ này.

var memoryBlockPointer = Marshal.AllocHGlobal(height \* stride);

// Copy các pixel của ảnh vào vùng nhớ không quản lý đã cấp phát ở trên.

// Vị trí đầu tiên của các pixel là tại vị trí (0,0) của ảnh (Int32Rect(0, 0, width, height)).

// Tham số thứ hai của hàm CopyPixels là con trỏ trỏ đến vùng nhớ không quản lý đã cấp phát ở trên.

// Tham số thứ ba là tổng kích thước của vùng nhớ đó (theo đơn vị byte).

// Tham số thứ tư là kích thước của mỗi hàng ảnh (theo đơn vị byte).

bitmapSource.CopyPixels(new Int32Rect(0, 0, width, height), memoryBlockPointer, height \* stride, stride);

// Tạo một đối tượng Bitmap mới với các thông số đã tính toán ở trên.

// Đối tượng Bitmap này được tạo từ vùng nhớ không quản lý đã copy pixel từ ảnh gốc.

// Lưu ý rằng đối tượng Bitmap được tạo ra này phải được giải phóng bằng cách gọi phương thức Dispose().

var bitmap = new Bitmap(width, height, stride, PixelFormat.Format32bppArgb, memoryBlockPointer);

// Trả về đối tượng Bitmap vừa tạo

return bitmap;

}

private string OCR(Bitmap b, string languageCode)// Phương thức này nhận đầu vào là một hình ảnh bitmap và trả về một chuỗi đã được nhận diện ký tự

{

string res = "";

// Tạo một đối tượng TesseractEngine để sử dụng thư viện OCR và đặt ngôn ngữ là tiếng Việt

using (var engine = new TesseractEngine(@"./tessdata", languageCode, EngineMode.Default))

{

// Sử dụng đối tượng TesseractEngine để xử lý hình ảnh và trả về một đối tượng Page

using (var page = engine.Process(b, PageSegMode.AutoOnly))

// Lấy chuỗi đã được nhận diện ký tự từ đối tượng Page và gán cho biến res

res = page.GetText();

}

// Trả về chuỗi đã được nhận diện ký tự

return res;

}

private void vietnameseToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

if (this.WindowState == FormWindowState.Normal)

{

// Thu nhỏ cửa sổ hiện tại để có thể chụp ảnh màn hình

this.WindowState = FormWindowState.Minimized;

// Dừng chương trình trong 0.5 giây để đợi cửa sổ thu nhỏ hoàn thành

Thread.Sleep(500);

// Chụp một ảnh vùng màn hình được chọn và chuyển đổi thành định dạng bitmap

}

try

{

Bitmap bitmap = ConvertToBitmap(Screenshot.CaptureRegion());

// Nhận diện ký tự trong ảnh bitmap sử dụng thư viện OCR và lưu kết quả vào biến ocrResult

string ocrResult = OCR(bitmap, "vie");

// Hiển thị kết quả nhận diện ký tự lên TextBox

tb\_quest.Text = ocrResult;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

// Khôi phục kích thước cửa sổ hiện tại

ShowHiddenForm(this);

}

### 4.1.7 Hàm tạo request API zalo.ai để phát âm thanh

public string GetAudioUrl(string text)

{

string apiKey = "f8p9cIrz7OXjS1Yi6dAQdce5FwQ30n1A";

string url = "https://api.zalo.ai/v1/tts/synthesize";

int speakerId = 4; // default speaker

float speed = 1.0f; // default speed

int quality = 0; // default quality

int encodeType = 0; // default encoding

string inputText = text;

// create request

var client = new RestClient(url);

var request = new RestRequest(Method.POST);

request.AddHeader("apikey", apiKey);

request.AddParameter("input", inputText);

request.AddParameter("speaker\_id", speakerId);

request.AddParameter("speed", speed);

request.AddParameter("quality", quality);

request.AddParameter("encode\_type", encodeType);

// execute request

IRestResponse response = client.Execute(request);

// check for errors

if (response.IsSuccessful)

{

var data = Newtonsoft.Json.JsonConvert.DeserializeObject<dynamic>(response.Content);

string audioUrl = data.data.url;

return audioUrl;

}

else

{

MessageBox.Show(response.Content, "API Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return null;

}

}

### 4.1.8 Hàm nhập bằng voice

private void cbtnVoiceInput\_Click\_1(object sender, EventArgs e)

{

SpeechRecognitionEngine recognier = new SpeechRecognitionEngine();//khởi tạo đối tượng nhân diện và xử lý ngôn ngữ nói

if (recognier.AudioState != AudioState.Silence)

{

recognier.SetInputToDefaultAudioDevice();//thiết lập thiết bị mặc định nhận âm thanh đầu vào

Grammar grammar = new DictationGrammar();// tạo đối tượng grammar thuộc lớp DictationGrammar - là một lớp con của lớp Grammar và đại diện cho một bộ từ điển cho phép người dùng nói bất kỳ từ hoặc cụm từ nào mà hệ thống nhận diện giọng nói có thể xử lý.

recognier.LoadGrammar(grammar);//tải bộ từ điển grammar vào đối tượng recognier để sử dụng cho việc nhận diện giọng nói.

recognier.SpeechRecognized += new EventHandler<SpeechRecognizedEventArgs>(recognier\_SpeechRecognized);//gán một phương thức xử lý sự kiện recognier\_SpeechRecognized vào sự kiện SpeechRecognized của đối tượng recognier. Phương thức này được gọi khi hệ thống nhận diện giọng nói thành công.

recognier.RecognizeAsync(RecognizeMode.Multiple);//bắt đầu một phiên nhận dạng giọng nói bằng cách sử dụng phương thức RecognizeAsync. Tham số RecognizeMode.Multiple được sử dụng để cho phép việc nhận dạng liên tục của nhiều cụm từ được phát hiện trong khi phiên nhận dạng giọng nói đang diễn ra.

}

if (recognier.AudioState == AudioState.Speech)

{

recognier.RecognizeAsyncStop(); // Dừng việc nhận diện giọng nói

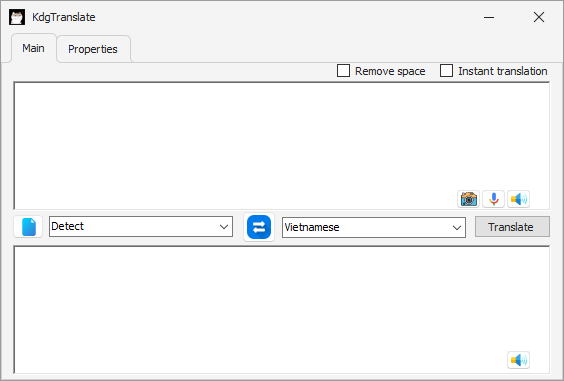
recognier.Dispose(); // Giải phóng tài nguyên

}

}

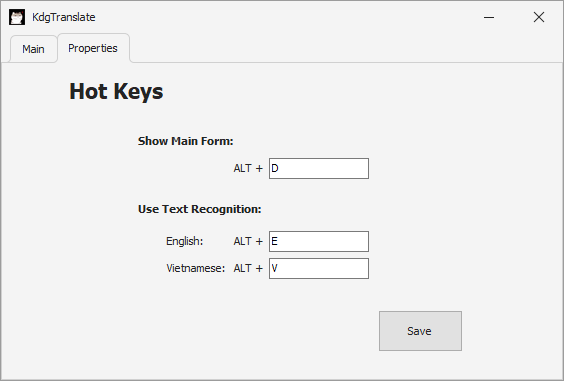
## 4.2. Kết quả lập trình

### 4.2.1 Main Form



### 4.2.2 Properties tab

\*Cài đặt phím tắt để sử dụng



# CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

### 5.1.1. Kết quả đạt được

Phần mềm phiên dịch đa ngôn ngữ sử dụng API của Google Translate và Zalo.ai đã hoàn thành các chức năng cơ bản như dịch văn bản, tự động phát hiện ngôn ngữ, nhập bằng giọng nói (tiếng Anh), đọc văn bản bản đa ngôn ngữ và ORC chụp và lấy văn bản từ ảnh (tiếng Anh - tiếng Việt). Phần mềm có giao diện đơn giản và dễ sử dụng, hỗ trợ 110 ngôn ngữ.

### 5.1.2 Hạn Chế

Tuy nhiên, phần mềm vẫn còn một số hạn chế như việc nhập bằng giọng nói chỉ hỗ trợ tiếng Anh và ORC chỉ hỗ trợ chụp và lấy văn bản từ ảnh đối với tiếng Anh và tiếng Việt. Ngoài ra, việc sử dụng API của Zalo.ai còn có thể gặp phải giới hạn về tốc độ và lượng dữ liệu được phép truy cập.

## 5.2. Hướng phát triển

Từ những hạn chế trên, cần có một hướng phát triển mới để đề tài ngày càng hoàn thiện hơn:

* Cải thiện chất lượng dịch thuật: Sử dụng các công nghệ dịch máy mới nhất và cải thiện thuật toán để đảm bảo chất lượng dịch thuật tốt hơn.
* Hỗ trợ nhiều ngôn ngữ trong tính năng voice input: Tăng cường khả năng nhận dạng giọng nói và hỗ trợ nhiều ngôn ngữ trong tính năng voice input.
* Mở rộng tính năng OCR: Phát triển tính năng OCR để hỗ trợ nhiều ngôn ngữ hơn, đảm bảo chính xác khi chụp và lấy văn bản từ ảnh.
* Tăng tốc độ xử lý: Tối ưu hóa thuật toán và sử dụng các công nghệ mới để tăng tốc độ xử lý và giảm thời gian chờ đợi của người dùng.
* Cải thiện hơn về trải nghiệm người dùng: Thêm những tính năng khác để tối ưu trải nghiệm sử dụng của người dùng

Tích hợp nhiều API dịch thuật khác: Tích hợp nhiều API dịch thuật khác nhau để tăng cường khả năng dịch thuật và đảm bảo sự linh hoạt trong việc sử dụng dịch vụ.

# TÀI LIỆU THAM KHẢO

<https://tuhoclaptrinh.edu.vn/bai-viet/lap-trinh-mang-la-gi-363.html>

<https://topdev.vn/blog/api-la-gi/>