**TESIS**

# PENERJEMAH BAHASA PEMROGRAMAN UNTUK INTEGRASI APLIKASI KLIEN DENGAN WEB APIS

# PROGRAMMING LANGUAGE TRANSLATOR FOR INTEGRATION CLIENT APPLICATION WITH WEB APIS

**MUDIARTA TAUDA**

**D032172008**


**TEKNIK ELEKTRO**

**FAKULTAS TEKNIK**

**UNIVERSITAS HASANUDDIN**

**MAKASSAR**

**2022**

Tesis

Sebagai Salah Satu Syarat Untuk Mencapai Gelar Magister

Program Studi

Teknik Elektro

Disusun dan diajukan oleh

**MUDIARTA TAUDA**

Kepada

**TEKNIK ELEKTRO**
**FAKULTAS TEKNIK**
**UNIVERSITAS HASANUDDIN**
**MAKASSAR**
**2022**

**LEMBAR PENGESAHAN TESIS**

Penerjemah Bahasa Pemrograman Untuk Integrasi Aplikasi Klien Dengan Web APIs

**Disusun dan diajukan oleh**

**MUDIARTA TAUDA**

**D032172008**

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Magister Program Studi Teknik Elektro Fakultas Teknik Universitas Hasanuddin
pada tanggal 16 Februari 2022
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

| Pembimbing Utama | Pembimbing Pendamping |
|---|---|
| **Dr. Ir. Zahir Zainuddin, M.Sc** | **Dr. Zulkifli Tahir, S.T., M.Sc** |
| **NIP. 196404271989101002** | **NIP. 198404032010121004** |

| Ketua Program Studi,<br>Magister Teknik Elektro | Dekan Fakultas Teknik,<br>Universitas Hasanuddin, |
|---|---|
| **Prof. Dr. Eng. Syafaruddin, S.T, M.Eng.** | **Prof. Dr. Eng. Muhammad Isran Ramli, ST.,M.T.** |
| **NIP. 197405301999031003** | **NIP. 197309262000121002** |

i

## PERNYATAAN KEASLIAN TESIS

Yang bertanda tangan dibawah ini:

| | |
|---|---|
| Nama | : Mudiarta Tauda |
| NIM | : D032172008 |
| Program Studi | :Teknik Elektro |
| Jenjang | : S2 |

Menyatakan dengan ini bahwa karya tulisan saya yang berjudul

Penerjemah Bahasa Pemrograman Untuk Integrasi Aplikasi Klien Dengan Web APIs

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain bahwa Tesis yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan Tesis ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan saya tersebut.

Makassar, 15 Februari 2022

Yang menyatakan,


Mudiarta Tauda

# KATA PENGANTAR

Segala puji kita panjatkan kehadirat Allah SWT. atas berkah dan rahmat-Nya sehingga penelitian dengan judul **“Penerjemah Bahasa Pemrograman Untuk Integrasi Aplikasi Klien Dengan Web Apis”** dapat diselesaikan.

Penulis menyadari bahwa banyak kendala yang dihadapi dalam menyelesaikan tesis ini, namun berkat dukungan, bimbingan dan bantuan berbagai pihak sehingga penyusunan tesis ini dapat terselesaikan dengan baik. Ucapan terima kasih dan penghargaan yang tak terhingga penulis sampaikan kepada semua pihak yang telah banyak mendukung dan membantu. Kepada kedua orang tua (Adam Tauda dan Maimuna Buamona),Ibu Angkat (Almh. Ratiah Buamona) dan saudara (Awaluddin Tauda) yang telah banyak mendukung dan bersabar dalam perjalanan studi penulis.

Kepada Prof. Dr. Eng. Syafaruddin, ST, M.Eng selaku ketua program studi S2 Teknik Elektro. Dr. Ir. Zahir Zainuddin, M.Sc., dan Dr-Eng. Zulkifli Tahir, S.T., M.Sc. selaku pembimbing yang senantiasa memberikan bimbingan, motivasi dan arahan-arahan dalam penyelesaian studi penulis. Kepada Amil Ahmad Ilham, ST., M.IT., Ph.D, Dr.Eng.Ir. Dewiani, M.T., Dr.

Hj. A. Ejah Urmaeni Salam, ST. MT, yang telah banyak memberikan kritik serta saran yang sifatnya membangun dalam penyusunan tesis ini. Bapak dan Ibu Dosen Program Studi Teknik Elektro FT UNHAS yang telah banyak memberikan bekal ilmu pengetahuan.

Untuk seluruh staf FT UNHAS, rekan-rekan seperjuangan di Pascasarjana angkatan 2017, teman-teman di laboratorium CBS, dan teman-teman terdekat penulis (Fatur, Kak Abe, Arfi, Fira) penulis juga mengucapkan terima kasih atas segala bantuan, dukungan dan kerja sama dalam menyelesaikan tesis ini.

Walaupun dalam penyusunannya, penulis telah berusaha dengan maksimal, namun jika masih ada kekurangan dari segi pengetikan maupun dari segi isi mohon kritik dan saran, demi penyusunan selanjutnya agar lebih baik lagi. Semoga tesis ini dapat menjadi suatu kontribusi bagi perkembangan ilmu pengetahuan.

Makassar, 15 Februari 2022

Yang menyatakan,

Mudiarta Tauda

## ABSTRAK

**Mudiarta Tauda**. Penerjemah Bahasa Pemrograman Untuk Integrasi Aplikasi Klien Dengan *Web APIs*. (dibimbing oleh Zahir Zainuddin dan Zulkifli Tahir)

Penelitian ini berdasarkan permasalahan yang ditemukan pada proses integrasi *Web API* yang sering mendapatkan masalah dikarenakan *Web API* yang diperbarui tidak lagi kompatibel dengan versi aplikasi klien. Untuk itu penelitian ini bertujuan mempermudah integrasi aplikasi klien berbasis android dengan *Web API*, dengan menerjemahkan *Web API* yang ditulis menggunakan bahasa pemrograman *typescript* ke aplikasi klien dengan bahasa pemrograman *kotlin* berbasis android. Dalam penelitian ini metode yang digunakan untuk proses penerjemahan *adalah Long Short-Term Memory (LSTM)* dengan pengaturan *learning rate* 0.003, hasil yang diperoleh dari penelitian ini adalah jaringan LSTM yang dihasilkan pada iterasi ke 20.000 mampu menerjemahkan 35 dari 102 pasang *keyword* dengan benar dan serta dapat menerjemahkan *Web API* kalkulator dan *translate* dengan benar. Sedangkan pada iterasi ke 40.000 LSTM mampu menerjemahkan 102 dari 102 pasang *keyword* dengan benar dan serta dapat menerjemahkan *Web API* kalkulator, translate, dan berita dengan benar.

Kata kunci: *REST API, Typescript, Kotlin, LSTM*

# ABSTRACT

**Mudiarta Tauda**. Programming Language Translator For Integration Client Application With Web APIs. (supervised by Ingrid Zahir Zainuddin dan Zulkifli Tahir)

This research is based on the problems found in the Web API integration process which often has problems because the updated Web API is no longer compatible with the client application version. For this reason, this study aims to facilitate the integration of Android-based client applications with Web APIs, by translating Web APIs written using the typescript programming language to client applications using the Android-based Kotlin programming language. In this study the method used for the translation process is Long Short-Term Memory (LSTM) with a learning rate setting of 0.003, the results obtained from this study are the LSTM network generated in the 20,000th iteration is able to translate 35 of 102 pairs of keywords correctly and can translate the calculator's Web API and translate correctly. Meanwhile, in the 40,000th iteration, LSTM was able to correctly translate 102 of 102 pairs of keywords and correctly translate the Web API calculator, translate, and news.

Keywords: REST API, Typescript, Kotlin, LSTM

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I
# PENDAHULUAN

## A. LATAR BELAKANG

Sistem komputer merupakan sekumpulan perangkat komputer yang saling berhubungan dan berinteraksi satu sama lain dengan tujuan untuk mengolah data. Fungsi dari sistem komputer antara lain: memasukkan data, memproses data, menghasilkan data, dan menyimpan data. Proses memasukkan data ke dalam sistem dari luar lingkungan sistem dapat dikatakan sebagai komputasi.

Komputasi berorientasi layanan merupakan salah satu jenis komputasi yang sekarang marak digunakan oleh perusahaan maupun organisasi untuk menyediakan layanan mereka melalui *web application programming interface (WEB APIs)*. Hal ini dibuktikan meningkatnya pertumbuhan *WEB APIs* dari 1 pada 2005 menjadi 14.667 pada 2016 (Pradeep K. Venkatesh et al., 2016).

*WEB APIs* modern sering mengikuti gaya arsitektur *Representational State Transfer (REST)* yang kemudian dikenal dengan istilah *RESTful WEB APIs* (Sergio Segura et al., 2017). Namun, proses integrasi *WEB APIs* sering kali mendapatkan masalah. Misalnya, *WEB APIs* yang diperbarui mungkin tidak lagi kompatibel dengan versi aplikasi klien saat ini, sehingga merusak aplikasi klien (Pradeep K. Venkatesh et al., 2016).

Sementara itu telah banyak penelitian yang dilakukan untuk penerjemah bahasa pemrograman, (Rahul Dubey et al., 2018) Mengusulkan *Algorithm To Code Converter* dengan tujuan untuk mengonversi algoritma yang ditulis dengan bahasa Inggris alami ke bahasa pemrograman *C* dan *Java* menggunakan metode klasifikasi *naïve bayes* dan *Part Of Speech (POS)*. (Xinyun Chen et al., 2018) Mengusulkan *Tree-to-tree Neural Networks for Program Translation*. Untuk menerjemahkan *Javascript ke Coffescript* dengan akurasi 75% dan untuk *Java* ke *C#* peningkatan 20,2%- 41,9%, dibandingkan beberapa metode lainya. (Mehdi Drissi et al., 2018) Mengusulkan Program *Language Translation Using a Grammar-Driven Tree-to-Tree Model*, mereka mengembangkan metode *Tree-to-tree Neural Networks for Program Translation Tree-to-tree Neural Networks for Program Translation* yang diusulkan oleh (Xinyun Chen et al., 2018) dengan memodifikasi *tree decode* untuk menggunakan tata bahasa target, Hasil dari penelitian ini adalah 85% akurasi untuk menerjemahkan *Javascript ke Coffescript*. (Trong Duc Nguyen et al., 2015) mengusulkan *Mapping API Elements for Code Migration with Vector Representations* untuk migrasi program yang dibuat menggunakan *Java* ke *C#* menggunakan metode *statistical approach with vector representations* akurasi yang dihasilkan dari metode yang diusulkan adalah 73,2%. (Yusuke Oda et al., 2015) mengusulkan *Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation* dengan tujuan untuk menerjemahkan program yang ditulis menggunakan Bahasa pemrograman *Python* ke

*pseudocode* dalam bahasa Inggris dan Jepang menggunakan metode *statistical machine translation (SMT)* akurasi yang dihasilkan 54,08% untuk *Python* ke Bahasa Inggris dan 62,88% untuk *Python* ke Jepang. (Anh Tuan Nguyen et al., 2015) mengusulkan *Divide-and-Conquer Approach for Multi-phase Statistical Migration for Source Code* untuk migrasi sistem yang dibuat menggunakan Bahasa pemrograman *Java* ke *C#* menggunakan metode *multi-phase phrase-based SMT (mppSMT)* akurasi yang dihasilkan untuk migrasi sistem 84.8 - 97.9%, migrasi metode sintaksis dan sematik 70–83%, migrasi metode yang sama persis dengan *C#* 26.3 - 51.2%. (Ebey Abraham et al., 2019) Mengusulkan *Real-Time Translation of Indian Sign Language using LSTM* metode yang digunakan adalah *Long Short Term Memory networks (LSTM)* sistem yang diusulkan mampu menerjemahkan Bahasa Isyarat India ke ucapan secara *real-time* dengan akurasi 98%.

Berdasarkan perkembangan teknologi saat ini serta masalah yang di kemukakan di atas, penulis melihat bahwa kebutuhan akan sebuah sistem yang mampu menerjemahkan *WEB APIs* ke aplikasi klien agar dapat diintegrasikan secara mudah saat ada perubahan dari *WEB APIs*. Untuk itu penulis mengajukan judul penelitian “penerjemah bahasa pemrograman untuk integrasi aplikasi klien dengan *WEB APIs*”. Penelitian ini diharapkan dapat membantu pengembang aplikasi klien untuk dapat dengan mudah mengintegrasi aplikasi yang dibuat dengan *WEB APIs* yang di gunakan.

## B. RUMUSAN MASALAH

Berdasarkan latar belakang masalah di atas maka rumusan masalah adalah bagaimana cara menerjemahkan *WEB APIs* yang dibuat menggunakan bahasa pemrograman *typescript* ke bahasa pemrograman kotlin, menggunakan metode *Long Short Term Memory Networks (LSTM)*.

## C. TUJUAN PENELITIAN

Berdasarkan rumusan masalah di atas maka tujuan dari penelitian ini adalah menghasilkan suatu sistem yang dapat menerjemahkan *source code* dari *WEB APIs* yang dibuat menggunakan Bahasa pemrograman typescript ke Bahasa pemrograman kotlin. agar dapat berkomunikasi menggunakan arsitektur *REST*.

## D. MANFAAT PENELITIAN

Berdasarkan tujuan penelitian di atas maka manfaat penelitian yang dapat diperoleh dari penelitian ini adalah mempermudah pengembang aplikasi klien untuk dapat mengintegrasikan aplikasi yang di buat dengan *web APIs.*

## E. BATASAN MASALAH

Batasan masalah dalam penelitian ini yaitu :

1. Penerjemahan dilakukan terhadap metode *GET, POST, PUT, DELETE dan Request Body*
2. *WEB APIs* dibuat menggunakan *framework Nest Js* dengan Bahasa Pemrograman *Typescript.*

3. *Source code* yang dihasilkan untuk aplikasi klien adalah *Kotlin* dan mengacu pada *library Retrofit* untuk integrasi ke *WEB APIs*.

## F. SISTEMATIKA PENULISAN

Adapun sistematika penulisan pada penelitian ini adalah

**Bab I Pendahuluan**

Bab I mencakup penjelasan tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah penelitian serta sistematika penulisan.

**Bab II Landasan Teori dan Kerangka Pemikiran**

Bab II penjelasan tentang landasan teori yang digunakan dalam penelitian dan kerangka pemikiran. Diuraikan tentang penelitian terkait yang berisi penjelasan tentang hasil-hasil penelitian yang menjadi acuan dengan penelitian yang akan dilakukan. Landasan teori seperti buku, 6 artikel, berita, jurnal, *prosiding*, dan tulisan asli lainnya untuk mengetahui perkembangan penelitian yang relevan dengan judul atau tema penelitian yang dilakukan dan juga sebagai arahan dalam memecahkan masalah yang diteliti. Kerangka pikir juga diuraikan dalam bab ini yang berisi penjelasan untuk memecahkan masalah yang sedang diteliti, termasuk menguraikan objek penelitian.

**Bab III Metodologi Penelitian**

Bab III berisi tentang metode penelitian, penentuan masalah, penentuan pendekatan komputasi, juga penjelasan bagaimana pengembangan dan penerapan sistem obyek penelitian. Selain itu bab III menjelaskan proses validasi hasil penerapan, metode pengumpulan data, metode analisis data, metode pengukuran penelitian, penerapan pada masalah penelitian, konstruksi sistem serta pengujian sistem.

**Bab IV Hasil dan Pembahasan**

Bab IV berisi penjelasan tentang hasil dan pembahasan penelitian serta implikasi dari penelitian yang dilakukan. Hasil penelitian berisi tentang hasil uji penerjemahan dalam bentuk tabel agar memudahkan pembaca dalam memahami hasil yang didapatkan. Pembahasan berisi penjelasan tentang pengolahan data dan interpretasinya, baik dalam bentuk deskriptif ataupun penarikan inferensinya. Implikasi Penelitian merupakan suatu penjelasan tentang tindak lanjut penelitian yang terkait dengan aspek sistem, maupun aspek penelitian lanjutan.

**Bab V Kesimpulan dan Saran**

Bab V peneliti menjelaskan ringkasan temuan, rangkuman, kesimpulan dan saran. Kesimpulan merupakan pernyataan secara umum atau spesifik yang berisi hal-hal penting dan menjadi temuan penelitian yang bersumber pada hasil dan pembahasan. Saran merupakan pernyataan atau rekomendasi peneliti yang berisi hal-hal penting yang perlu dilakukan pada penelitian selanjutnya.

# BAB II
# TINJAUAN PUSTAKA

## A. LANDASAN TEORI

### *1. RESTful WEB APIs*

API adalah sebuah dokumentasi yang terdiri dari *interface*, kelas, fungsi, struktur dan sebagainya agar dapat membangun sebuah aplikasi, API juga dapat dikatakan sebagai suatu kode pemrograman penghubung antara aplikas (Untung Rahardja., dkk. 2018). API memungkinkan developer untuk mengintegrasikan dan mengizinkan dua aplikasi saling berkomunikasi untuk mempermudah pengembangan aplikasi, tujuan penggunaan API yaitu untuk mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah *function* yang terpisah sehingga para developer tidak perlu lagi membuat fitur yang serupa.

*Representational State Transfer* (REST) merupakan arsitektur dalam mendesain sebuah *web service* di mana desain REST memiliki sumber daya yang dapat diakses melalui sebuah alamat HTTP URL yang unik. REST juga memungkinkan klien dapat melakukan permintaan melalui protokol HTTP dengan mudah menggunakan URI (Muhammad Iqbal Perkasa., dkk. 2018).

*WEB APIs* yang mematuhi batasan arsitektur *REST* disebut *RESTful WEB APIs*. *RESTful WEB APIs* di dekomposisi menjadi beberapa layanan *WEB APIs*, di mana setiap layanan mengimplementasikan satu atau lebih

operasi *CRUD* di atas sumber daya. Sumber daya adalah segala sesuatu yang dapat diekspos ke *WEB APIs* seperti video, foto, atau teks (Sergio Segura., dkk. 2017). Dengan menggunakan arsitektur *RESTful WEB APIs* maka proses integrasi antara dua buah aplikasi tidak hanya melalui Bahasa pemrograman yang sama, dengan menggunakan arsitektur *RESTful WEB APIs* maka dua buah aplikasi dapat diintegrasikan melalui *protocol HTTP*.

Dengan menggunakan arsitektur REST maka aplikasi klien dapat mengirimkan perintah yang akan dikerjakan oleh server menggunakan metode-metode HTTP *request method* yang disebut *verb*. terdapat delapan *HTTP request method*, yaitu *GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, dan CONNECT*. Dalam penggunaan *API REST* hanya menggunakan empat dari metode-metode tersebut, yaitu: *GET, POST, PUT, dan DELETE*, (Muhammad Iqbal Perkasa., dkk. 2018). Sementara itu dalam memberikan *response* terhadap permintaan yang dikirimkan oleh aplikasi klien maka aplikasi server akan memberikan *response* berupa *http status code* pada *header* serta hasil dari permintaan tersebut dalam *response body*. Adapun *http status code* berdasarkan standar yang dirilis pada www.w3.org adalah sebagai berikut.

**Tabel 2.1** *Http status code*

| Code | Deskripsi |
|---|---|
| 100 | Klien Harus melanjutkan permintaannya. Respons sementara ini digunakan untuk menginformasikan klien bahwa bagian awal dari permintaan telah diterima dan belum ditolak oleh server. Klien Harus melanjutkan dengan mengirimkan sisa permintaan atau, jika permintaan telah selesai, abaikan respons ini. Server HARUS |

| | |
|---|---|
| | mengirim tanggapan akhir setelah permintaan selesai |
| 101 | Server memahami dan bersedia memenuhi permintaan klien, melalui bidang header pesan Upgrade, untuk perubahan dalam protokol aplikasi yang digunakan pada koneksi ini. Server akan mengalihkan protokol ke protokol yang ditentukan oleh bidang tajuk Upgrade respons segera setelah baris kosong yang mengakhiri respons 101. |
| 200 | Permintaan telah berhasil. Informasi yang dikembalikan dengan respons bergantung pada metode yang digunakan dalam permintaan, misalnya:<br><br>GET entitas yang sesuai dengan sumber daya yang diminta dikirim dalam respons;<br><br>HEAD bidang entitas-header yang sesuai dengan sumber daya yang diminta dikirim dalam respons tanpa isi pesan apa pun;<br><br>POST entitas yang menjelaskan atau berisi hasil tindakan;<br><br>TRACE entitas yang berisi pesan permintaan seperti yang diterima oleh server akhir. |
| 201 | Permintaan telah dipenuhi dan menghasilkan sumber daya baru yang dibuat. Sumber daya yang baru dibuat dapat direferensikan oleh URI yang dikembalikan dalam entitas respons. |
| 202 | Permintaan telah diterima untuk diproses, tetapi pemrosesan belum selesai. Permintaan mungkin atau mungkin tidak pada akhirnya ditindaklanjuti, karena mungkin tidak diizinkan saat pemrosesan benar-benar terjadi. Tidak ada fasilitas untuk mengirim ulang kode status dari operasi asinkron seperti ini.<br><br>Tanggapan 202 sengaja tidak berkomitmen. Tujuannya adalah untuk memungkinkan server menerima permintaan untuk beberapa proses lain (mungkin proses berorientasi batch yang hanya dijalankan sekali per hari) tanpa mengharuskan koneksi agen pengguna ke server bertahan sampai proses selesai. Entitas yang dikembalikan dengan respons ini harus menyertakan indikasi status permintaan saat ini dan penunjuk ke monitor status atau perkiraan kapan pengguna dapat mengharapkan permintaan dipenuhi. |

| | |
|---|---|
| 203 | Metainformasi yang dikembalikan di header entitas bukanlah kumpulan definitif yang tersedia dari server asal, tetapi dikumpulkan dari salinan lokal atau pihak ketiga. Himpunan yang disajikan mungkin menjadi subset atau superset dari versi aslinya. Misalnya, menyertakan informasi anotasi lokal tentang sumber daya dapat menghasilkan superset dari metainformasi yang diketahui oleh server asal. Penggunaan kode respons ini tidak diperlukan dan hanya sesuai bila respons sebaliknya adalah 200 (OK). |
| 204 | Server telah memenuhi permintaan tetapi tidak perlu mengembalikan badan entitas, dan mungkin ingin mengembalikan informasi meta yang diperbarui. Tanggapan MUNGKIN menyertakan metainformasi baru atau yang diperbarui dalam bentuk header entitas, yang jika ada HARUS dikaitkan dengan varian yang diminta.<br><br>Jika klien adalah agen pengguna, itu TIDAK HARUS mengubah tampilan dokumennya dari yang menyebabkan permintaan dikirim. Respons ini terutama dimaksudkan untuk memungkinkan masukan untuk tindakan yang terjadi tanpa menyebabkan perubahan pada tampilan dokumen aktif agen pengguna, meskipun informasi meta baru atau yang diperbarui HARUS diterapkan ke dokumen yang saat ini berada dalam tampilan aktif agen pengguna.<br><br>Respons 204 TIDAK HARUS menyertakan badan pesan, dan dengan demikian selalu diakhiri oleh baris kosong pertama setelah bidang header. |
| 205 | Server telah memenuhi permintaan dan agen pengguna HARUS mengatur ulang tampilan dokumen yang menyebabkan permintaan dikirim. Respons ini terutama dimaksudkan untuk memungkinkan input untuk tindakan dilakukan melalui input pengguna, diikuti dengan pembersihan formulir di mana input diberikan sehingga pengguna dapat dengan mudah memulai tindakan input lain. Respons TIDAK HARUS menyertakan entitas. |
| 206 | Server telah memenuhi sebagian permintaan GET untuk sumber daya. Permintaan HARUS menyertakan bidang header Rentang yang menunjukkan rentang yang diinginkan, dan MUNGKIN telah menyertakan bidang header If-Range untuk membuat permintaan bersyarat. |

| | |
|---|---|
| | Respons HARUS menyertakan bidang tajuk berikut:<br>✓ Bidang header Rentang Konten menunjukkan rentang yang disertakan dengan respons ini, atau multipart/byteranges Content-Type termasuk bidang Content-Range untuk setiap bagian. Jika sebuah Bidang tajuk Panjang Konten hadir dalam respons, itu nilai HARUS cocok dengan jumlah sebenarnya dari OCTET yang ditransmisikan dalam Badan Pesan.<br>✓ Tanggal<br>✓ ETag dan/atau Content-Location, jika header akan dikirim dalam tanggapan 200 untuk permintaan yang sama<br>✓ Kedaluwarsa, Kontrol Cache, dan/atau Bervariasi, jika nilai bidang mungkin berbeda dari yang dikirim dalam tanggapan sebelumnya untuk hal yang sama varian<br><br>Jika respons 206 adalah hasil dari permintaan If-Range yang menggunakan validator cache yang kuat, respons TIDAK HARUS menyertakan header entitas lainnya. Jika respons adalah hasil dari permintaan If-Range yang menggunakan validator yang lemah, respons HARUS TIDAK menyertakan header entitas lainnya; ini mencegah inkonsistensi antara badan entitas yang di-cache dan header yang diperbarui. Jika tidak, respons HARUS menyertakan semua header entitas yang akan dikembalikan dengan respons 200 (OK) untuk permintaan yang sama. |
| 300 | Sumber daya yang diminta sesuai dengan salah satu dari satu set representasi, masing-masing dengan lokasi spesifiknya sendiri, dan informasi negosiasi yang digerakkan oleh agen disediakan sehingga pengguna (atau agen pengguna) dapat memilih representasi yang disukai dan mengarahkan ulangnya. permintaan ke lokasi itu.<br><br>Kecuali jika permintaan HEAD, respons HARUS menyertakan entitas yang berisi daftar karakteristik sumber daya dan lokasi dari mana pengguna atau agen pengguna dapat memilih salah satu yang paling sesuai. Format entitas ditentukan oleh jenis media yang diberikan di bidang header Jenis Konten. Tergantung pada format dan kemampuan<br><br>agen pengguna, pemilihan pilihan yang paling tepat MUNGKIN dilakukan secara otomatis. Namun, spesifikasi ini tidak menentukan standar apa pun untuk pemilihan otomatis tersebut.<br><br>Jika server memiliki pilihan representasi yang lebih disukai, |

| | |
|---|---|
| | server HARUS menyertakan URI spesifik untuk representasi tersebut di bidang Lokasi; agen pengguna MUNGKIN menggunakan nilai bidang Lokasi untuk pengalihan otomatis. Tanggapan ini dapat disimpan dalam cache kecuali dinyatakan lain. |
| 301 | Sumber daya yang diminta telah diberi URI permanen baru dan referensi di masa mendatang untuk sumber daya ini HARUS menggunakan salah satu URI yang dikembalikan. Klien dengan kemampuan pengeditan tautan harus secara otomatis menautkan ulang referensi ke Request-URI ke satu atau lebih referensi baru yang dikembalikan oleh server, jika memungkinkan. Tanggapan ini dapat disimpan dalam cache kecuali dinyatakan lain.<br><br>URI permanen baru HARUS diberikan oleh bidang Lokasi dalam respons. Kecuali jika metode permintaan adalah HEAD, entitas respons HARUS berisi catatan hypertext pendek dengan hyperlink ke URI baru.<br><br>Jika kode status 301 diterima sebagai tanggapan atas permintaan selain GET atau HEAD, agen pengguna TIDAK HARUS secara otomatis mengalihkan permintaan kecuali dapat dikonfirmasi oleh pengguna, karena ini dapat mengubah kondisi di mana permintaan dikeluarkan. |
| 302 | Sumber daya yang diminta berada sementara di bawah URI yang berbeda. Karena pengalihan mungkin berubah sewaktu-waktu, klien HARUS terus menggunakan Request-URI untuk permintaan di masa mendatang. Respons ini hanya dapat disimpan dalam cache jika ditunjukkan oleh bidang header Kontrol Cache atau Kedaluwarsa.<br><br>URI sementara HARUS diberikan oleh bidang Lokasi dalam respons. Kecuali jika metode permintaan adalah HEAD, entitas respons HARUS berisi catatan hypertext pendek dengan hyperlink ke URI baru.<br><br>Jika kode status 302 diterima sebagai tanggapan atas permintaan selain GET atau HEAD, agen pengguna TIDAK HARUS secara otomatis mengalihkan permintaan kecuali dapat dikonfirmasi oleh pengguna, karena ini dapat mengubah kondisi di mana permintaan tersebut dikeluarkan. |

| | |
|---|---|
| 303 | Respons terhadap permintaan dapat ditemukan di bawah URI yang berbeda dan HARUS diambil menggunakan metode GET pada sumber daya itu. Metode ini ada terutama untuk memungkinkan keluaran skrip yang diaktifkan POST untuk mengarahkan ulang agen pengguna ke sumber daya yang dipilih. URI baru bukan referensi pengganti untuk sumber daya yang awalnya diminta. Respons 303 TIDAK HARUS di-cache, tetapi respons terhadap permintaan kedua (dialihkan) mungkin dapat di-cache.<br><br>URI yang berbeda HARUS diberikan oleh bidang Lokasi dalam respons. Kecuali jika metode permintaan adalah HEAD, entitas respons HARUS berisi catatan hypertext pendek dengan hyperlink ke URI baru. |
| 304 | Jika klien telah melakukan permintaan GET bersyarat dan akses diizinkan, tetapi dokumen belum diubah, server HARUS merespons dengan kode status ini. Respons 304 HARUS TIDAK berisi badan pesan, dan karenanya selalu diakhiri oleh baris kosong pertama setelah bidang header. |
| 305 | Sumber daya yang diminta HARUS diakses melalui proxy yang diberikan oleh bidang Lokasi. Bidang Lokasi memberikan URI proxy. Penerima diharapkan untuk mengulangi permintaan tunggal ini melalui proxy. 305 tanggapan HARUS hanya dihasilkan oleh server asal. |
| 306 | Kode status 306 digunakan dalam versi spesifikasi sebelumnya, tidak lagi digunakan, dan kode dicadangkan. |
| 307 | Sumber daya yang diminta berada sementara di bawah URI yang berbeda. Karena pengalihan DAPAT diubah sesekali, klien HARUS terus menggunakan Request-URI untuk permintaan di masa mendatang. Respons ini hanya dapat disimpan dalam cache jika ditunjukkan oleh bidang header Kontrol Cache atau Kedaluwarsa.<br><br>URI sementara HARUS diberikan oleh bidang Lokasi dalam respons. Kecuali jika metode permintaan adalah HEAD, entitas respons HARUS berisi catatan hypertext pendek dengan hyperlink ke URI baru, karena banyak agen pengguna pra-HTTP/1.1 tidak memahami status 307. Oleh karena itu, catatan HARUS berisi informasi yang diperlukan pengguna untuk mengulangi permintaan asli pada URI baru. |

| | |
|---|---|
| | Jika kode status 307 diterima sebagai tanggapan atas permintaan selain GET atau HEAD, agen pengguna TIDAK HARUS secara otomatis mengarahkan permintaan kecuali dapat dikonfirmasi oleh pengguna, karena ini dapat mengubah kondisi di mana permintaan dikeluarkan. |
| 400 | Permintaan tidak dapat dipahami oleh server karena sintaks yang salah. Klien TIDAK HARUS mengulangi permintaan tanpa modifikasi. |
| 401 | Permintaan memerlukan otentikasi pengguna. |
| 402 | Kode ini dicadangkan untuk penggunaan di masa mendatang. |
| 403 | Server memahami permintaan tersebut, tetapi menolak untuk memenuhinya. Otorisasi tidak akan membantu dan permintaan TIDAK HARUS diulang. Jika metode permintaan bukan HEAD dan server ingin mengumumkan mengapa permintaan tidak terpenuhi, HARUS menjelaskan alasan penolakan dalam entitas. Jika server tidak ingin membuat informasi ini tersedia untuk klien, kode status 404 (Tidak Ditemukan) dapat digunakan sebagai gantinya. |
| 404 | Server belum menemukan apa pun yang cocok dengan Request-URI. Tidak ada indikasi yang diberikan apakah kondisi tersebut bersifat sementara atau permanen. Kode status 410 (Hilang) HARUS digunakan jika server mengetahui, melalui beberapa mekanisme yang dapat dikonfigurasi secara internal, bahwa sumber daya lama tidak tersedia secara permanen dan tidak memiliki alamat penerusan. Kode status ini biasanya digunakan ketika server tidak ingin mengungkapkan dengan tepat mengapa permintaan ditolak, atau ketika tidak ada respons lain yang berlaku. |
| 405 | Metode yang ditentukan dalam Request-Line tidak diperbolehkan untuk sumber daya yang diidentifikasi oleh Request-URI. Respons HARUS menyertakan header Allow yang berisi daftar metode yang valid untuk resource yang diminta. |
| 406 | Sumber daya yang diidentifikasi oleh permintaan hanya mampu menghasilkan entitas respons yang memiliki karakteristik konten yang tidak dapat diterima sesuai dengan header penerimaan yang dikirim dalam permintaan.<br><br>Kecuali jika permintaan HEAD, respons HARUS menyertakan entitas yang berisi daftar karakteristik entitas yang tersedia dan lokasi dari mana pengguna atau agen pengguna dapat memilih salah satu yang paling sesuai. Format entitas ditentukan oleh |

| | |
|---|---|
| | jenis media yang diberikan di bidang header Jenis Konten. Tergantung pada format dan kemampuan agen pengguna, pemilihan pilihan yang paling tepat MUNGKIN dilakukan secara otomatis. Namun, spesifikasi ini tidak menentukan standar apa pun untuk pemilihan otomatis tersebut. |
| 407 | Kode ini mirip dengan 401 (Tidak Diotorisasi), tetapi menunjukkan bahwa klien harus terlebih dahulu mengotentikasi dirinya dengan proxy. Proxy HARUS mengembalikan bidang header Proxy-Authenticate yang berisi tantangan yang berlaku untuk proxy untuk sumber daya yang diminta. Klien MUNGKIN mengulangi permintaan dengan bidang header Proxy-Otorisasi yang sesuai. Otentikasi akses HTTP dijelaskan dalam "Otentikasi HTTP: Otentikasi Akses Dasar dan Intisari" |
| 408 | Klien tidak menghasilkan permintaan dalam waktu server siap untuk menunggu. Klien MUNGKIN mengulangi permintaan tanpa modifikasi di lain waktu. |
| 409 | Permintaan tidak dapat diselesaikan karena konflik dengan status sumber daya saat ini. Kode ini hanya diperbolehkan dalam situasi di mana diharapkan pengguna dapat menyelesaikan konflik dan mengirim ulang permintaan. Badan respons HARUS menyertakan cukup<br><br>informasi bagi pengguna untuk mengenali sumber konflik. Idealnya, entitas respons akan menyertakan informasi yang cukup bagi pengguna atau agen pengguna untuk memperbaiki masalah; namun, itu mungkin tidak mungkin dan tidak diperlukan.<br><br>Konflik kemungkinan besar terjadi sebagai tanggapan atas permintaan PUT. Misalnya, jika pembuatan versi sedang digunakan dan entitas yang PUT menyertakan perubahan pada sumber daya yang bertentangan dengan yang dibuat oleh permintaan (pihak ketiga) sebelumnya, server mungkin menggunakan respons 409 untuk menunjukkan bahwa ia tidak dapat menyelesaikan permintaan . Dalam hal ini, entitas respons kemungkinan akan berisi daftar perbedaan antara dua versi dalam format yang ditentukan oleh Tipe-Konten respons. |
| 410 | Sumber daya yang diminta tidak lagi tersedia di server dan tidak ada alamat penerusan yang diketahui. Kondisi ini diperkirakan akan dianggap permanen. Klien dengan kemampuan pengeditan tautan HARUS menghapus referensi ke Request-URI setelah persetujuan pengguna. Jika server tidak mengetahui, atau tidak memiliki fasilitas untuk menentukan, apakah kondisi tersebut |

| | |
|---|---|
| | permanen atau tidak, kode status 404 (Not Found) HARUS digunakan sebagai gantinya. Tanggapan ini dapat disimpan dalam cache kecuali dinyatakan lain.<br><br>Respons 410 terutama ditujukan untuk membantu tugas pemeliharaan web dengan memberi tahu penerima bahwa sumber daya sengaja tidak tersedia dan bahwa pemilik server menginginkan agar tautan jarak jauh ke sumber daya tersebut dihapus. Peristiwa seperti itu biasa terjadi untuk layanan promosi dengan waktu terbatas dan untuk sumber daya milik individu yang tidak lagi bekerja di situs server. Tidak perlu menandai semua sumber daya yang tidak tersedia secara permanen sebagai "hilang" atau mempertahankan tanda untuk waktu yang lama yang diserahkan kepada kebijaksanaan pemilik server. |
| 411 | Server menolak untuk menerima permintaan tanpa Panjang Konten yang ditentukan. Klien MUNGKIN mengulangi permintaan jika menambahkan bidang header Panjang Konten yang valid yang berisi panjang badan pesan dalam pesan permintaan. |
| 412 | Prakondisi yang diberikan dalam satu atau beberapa bidang header permintaan yang dievaluasi salah saat diuji di server. Kode respons ini memungkinkan klien untuk menempatkan prasyarat pada metainformasi sumber daya saat ini (data bidang header) dan dengan demikian mencegah metode yang diminta diterapkan ke sumber daya selain yang dimaksudkan. |
| 413 | Server menolak untuk memproses permintaan karena entitas permintaan lebih besar dari yang ingin atau mampu diproses oleh server. Server MUNGKIN menutup koneksi untuk mencegah klien melanjutkan permintaan.<br><br>Jika kondisinya sementara, server HARUS menyertakan bidang tajuk Coba Lagi- Setelah untuk menunjukkan bahwa itu sementara dan setelah jam berapa klien MUNGKIN mencoba lagi. |
| 414 | Server menolak untuk melayani permintaan karena Request-URI lebih panjang dari yang bersedia diinterpretasikan oleh server. Kondisi langka ini hanya mungkin terjadi saat klien salah mengonversi permintaan POST ke permintaan GET dengan informasi kueri panjang, saat klien turun ke "lubang hitam" pengalihan URI (misalnya, awalan URI yang dialihkan yang menunjuk ke akhiran itu sendiri), atau ketika server diserang oleh klien yang mencoba mengeksploitasi lubang keamanan yang ada di beberapa server menggunakan buffer dengan panjang tetap |

| | |
|---|---|
| | untuk membaca atau memanipulasi Request-URI. |
| 415 | Server menolak untuk melayani permintaan karena entitas permintaan dalam format yang tidak didukung oleh sumber daya yang diminta untuk metode yang diminta. |
| 416 | Server HARUS mengembalikan respons dengan kode status ini jika permintaan menyertakan bidang header permintaan Rentang dan tidak ada nilai penentu rentang di bidang ini yang tumpang tindih dengan tingkat sumber daya yang dipilih saat ini, dan permintaan tidak sertakan bidang header permintaan If-Range. (Untuk rentang-byte, ini berarti bahwa pos-byte pertama dari semua nilai spesifikasi rentang-byte lebih besar dari panjang sumber daya yang dipilih saat ini.)<br><br>Saat kode status ini dikembalikan untuk permintaan rentang byte, respons HARUS menyertakan bidang header entitas Rentang Konten yang menentukan panjang saat ini dari sumber daya yang dipilih. Respons ini TIDAK HARUS menggunakan tipe konten multipart/byteranges. |
| 417 | Ekspektasi yang diberikan dalam bidang header permintaan Harapkan (lihat bagian 14.20) tidak dapat dipenuhi oleh server ini, atau, jika server adalah proxy, server memiliki bukti yang jelas bahwa permintaan tersebut tidak dapat dipenuhi oleh server hop berikutnya . |
| 500 | Server mengalami kondisi tak terduga yang mencegahnya memenuhi permintaan. |
| 501 | Server tidak mendukung fungsionalitas yang diperlukan untuk memenuhi permintaan. Ini adalah respons yang tepat ketika server tidak mengenali metode permintaan dan tidak mampu mendukungnya untuk sumber daya apa pun. |
| 502 | Server, saat bertindak sebagai gateway atau proxy, menerima respons yang tidak valid dari server upstream yang diaksesnya dalam upaya memenuhi permintaan. |
| 503 | Server saat ini tidak dapat menangani permintaan karena kelebihan beban sementara atau pemeliharaan server. Implikasinya adalah bahwa ini adalah kondisi sementara yang akan berkurang setelah beberapa penundaan. Jika diketahui, panjang penundaan MUNGKIN ditunjukkan dalam header Coba Lagi-Setelah. Jika tidak ada Retry-After yang diberikan, klien HARUS menangani respons seperti halnya untuk respons 500. |
| 504 | Server, saat bertindak sebagai gateway atau proxy, tidak menerima respons tepat waktu dari server upstream yang |

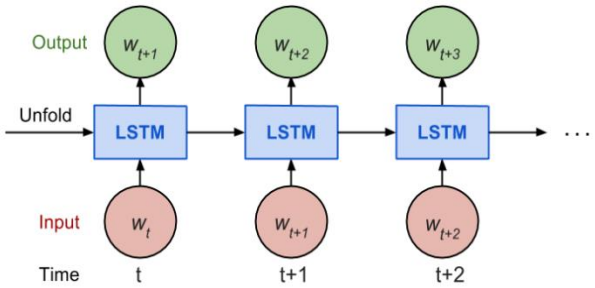
| | |
|---|---|
| | ditentukan oleh URI (misalnya HTTP, FTP, LDAP) atau beberapa server tambahan lainnya (misalnya DNS) yang diperlukan untuk mengakses dalam upaya menyelesaikan Permintaan. |
| 505 | Server tidak mendukung, atau menolak mendukung, versi protokol HTTP yang digunakan dalam pesan permintaan. Server menunjukkan bahwa ia tidak dapat atau tidak mau menyelesaikan permintaan menggunakan versi utama yang sama seperti klien, selain dengan pesan kesalahan ini. Respons HARUS berisi entitas yang menjelaskan mengapa versi itu tidak didukung dan protokol lain apa yang didukung oleh server itu. |

*2. Long Short Term Memory Networks (LSTM)*

Long Short Term Memory networks (LSTM) merupakan sebuah evolusi dari arsitektur RNN, dimana pertama kali diperkenalkan oleh Hochreiter & Schmidhuber (1997) (Muhammad Wildan Putra Aldi*.,* dkk, 2018*). Long Short Term Memory* (LSTM) merupakan salah satu pengembangan neural network yang dapat digunakan untuk pemodelan data time series. LSTM mampu mengatasi ketergantungan jangka panjang (long term dependencies) pada masukannya. Sebuah penelitian *A New Method for Semantic Consistency Verification of Aviation Radiotelephony Communication Based on* LSTM-RNN mengatakan bahwa LSTM berhasil diterapkan pada berbagai tugas sekuensial dan bahasa pemodelan. Sebuah cell dalam LSTM menyimpan sebuah nilai atau keadaan (cell state), baik untuk periode waktu yang panjang atau singkat. LSTM mempunyai memory block yang akan menentukan nilai mana yang akan dipilih sebagai keluaran yang relevan terhadap masukan yang diberikan. Hal ini adalah keunggulan yang dimiliki oleh LSTM. Pada penelitian ini dilakukan 3 tahap utama yaitu: 1) *Preprocessing Data*; 2) Pembuatan model melalui proses

training LSTM Network; 3) Melakukan uji terhadap data testing( Laras Wiranda., dkk, 2019).

LSTM disebut juga sebagai jaringan saraf dengan arsitektur yang mudah beradaptasi, sehingga bentuknya dapat disesuaikan, tergantung pada aplikasinya. Long Short Term Memory merupakan turunan dari metode RNN (Recurrent Neural Network). Recurrent Neural Network merupakan jaringan saraf berulang yang didesain khusus untuk menghandle data berurutan (sequence data). Namun RNN mempunyai masalah *vanishing* dan *exploding gradient* yaitu apabila terjadi perubahan pada jangkauan nilai dari satu lapisan menuju lapisan berikutnya pada sebuah arsitektur. LSTM dibangun dan dirancang untuk mengatasi masalah gradien menghilang dari RNN ketika berhadapan dengan vanishing dan exploding gradient tersebut. Gambar dibawah ini merupakan model dari sell LSTM.




**Gambar 2.1** Arsitektur LSTM

## B. PENELITIAN TERKAIT

Berikut adalah beberapa penelitian terkait baik itu dari *literature* jurnal internasional maupun nasional yang menjadi acuan untuk pengembangan ke depannya.

(Anh Tuan Nguye et al., 2016) Mengusulkan *Divide-and-Conquer Approach for Multi-phase Statistical Migration for Source Code*. Untuk mengatasi keterbatasan utama penggunaan urutan dalam SMT berbasis frase dalam memodelkan dan menerjemahkan kode sumber dengan struktur yang terbentuk dengan baik. Dalam penelitian ini mereka mengusulkan *mppSMT* yang menggunakan teknik *divide-and-conquer* yang terbagi dalam tiga fase. Pertama, *mppSMT* memperlakukan program sebagai urutan unit sintaksis dan memetakan / menerjemahkan urutan tersebut dalam dua bahasa satu sama lain. Kedua, mode yang diarahkan sintaksis, berurusan dengan *token* dalam unit sintaksis dengan *mengodekannya* dengan simbol semantik untuk mewakili data dan tipe *token*. Ketiga, *token* leksikal yang sesuai dengan setiap bahasa target dipetakan. Hasilnya migrasi sistem 84.8 - 97.9%, migrasi metode sintaksis dan sematik 70–83%, migrasi metode yang sama persis dengan *C#* 26.3 - 51.2%.

(Trong Duc Nguyen et al., 2016) Mengusulkan *Mapping API Elements for Code Migration with Vector Representations*, tujuan dari penelitian ini adalah memaksimalkan migrasi API secara otomatis menggunakan teknik penambangan. penelitian ini menggunakan pendekatan untuk secara

otomatis menambang pemetaan API untuk mengatasi masalah ketidakcocokan leksikal. Penelitian ini menggunakan word2vec untuk memproyeksikan API Java JDK dan C # .NET ke dalam ruang *vector* berkelanjutan yang sesuai. Hasilnya metode yang diusulkan dapat memperoleh API dengan benar di C # di hampir 43% kasus. Dengan 5 saran, metode yang diusulkan dapat dengan benar menyarankan C # API di hampir 3 dari 4 kasus (73,2%).

(Yusuke Oda et al., 2015) Mengusulkan *Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation*. Tujuan dari penelitian ini adalah mempermudah proses pembuatan *pseudocode*. Penelitian ini menggunakan *statistical machine translation (SMT)* untuk menghasilkan *pseudocode* dari sebuah program yang telah dibuat menggunakan *python* ke dalam bahasa Inggris dan Jepang. Akurasi dari metode yang diusulkan dalam penelitian ini adalah BLEU *score python* ke Inggris 54.08% dan BLEU *score python* ke Jepang: 62.88%.

(Xinyun Chen et al., 2018) Mengusulkan *Tree-to-tree Neural Networks for Program Translation*. Tujuan dari penelitian ini adalah untuk menerjemahkan program yang dibuat menggunakan *java* ke *c#, coffescript* ke *javascript*. dan sebaliknya. Penelitian ini menggunakan pendekatan *neural machine translation* dengan memanfaatkan model *encode/decoder tree*, hasilnya *tree-to-tree* unggul 75% untuk *coffescript* ke *javascript* dan sebaliknya dibandingkan dengan *seq2seq, seq2tree, tree2seq*. untuk *java* ke *c#* dan sebaliknya peningkatan 20,2%- 41,9%. Dibandingkan *J2C#.*

*lpSMT, mppSMT*.

(Mehdi Drissi et al., 2018) Mengusulkan Program *Language Translation Using a Grammar-Driven Tree-to-Tree Model*. Penelitian ini menerapkan model *encoder / decoder tree-to-tree* (Chen., dkk., 2018). Yang membedakan adalah mereka memodifikasi *tree decoder* untuk menggunakan tata bahasa bahasa target saat membuat node. Mereka tidak membuat *binary output tree*, karena aturan tata bahasa bahasa target tidak dapat dengan mudah diterapkan pada pohon di mana saudara simpul dapat muncul sebagai anak-anaknya. Hasil dari penelitian ini adalah 85% akurasi.

(Ebey Abraham et al., 2019) Mengusulkan Real-Time Translation of Indian Sign Language using LSTM yang mampu menerjemahkan Bahasa Isyarat India ke ucapan secara real-time. Model yang diusulkan dilatih untuk 26 kata dalam Bahasa Isyarat India dengan 40 sampel untuk setiap kata. Model yang dilatih dapat mencapai akurasi 98%.

## C. STATE OF THE ART

Berikut beberapa penelitian tentang sistem penerjemah bahasa pemrograman. yang telah dilakukan dan yang akan diusulkan dapat dilihat pada tabel 1 *state of the art*. Penelitian tersebut telah dirancang untuk membantu pengembangan untuk migrasi lingkungan sistem dari satu bahasa pemrograman ke bahasa pemrograman lain. Namun, penelitian yang telah dilakukan belum menerapkan sistem penerjemah bahasa pemrograman tersebut ke dalam integrasi *APIs*, sehingga diusulkan

penerjemah bahasa pemrograman untuk integrasi aplikasi klien dengan web apis.

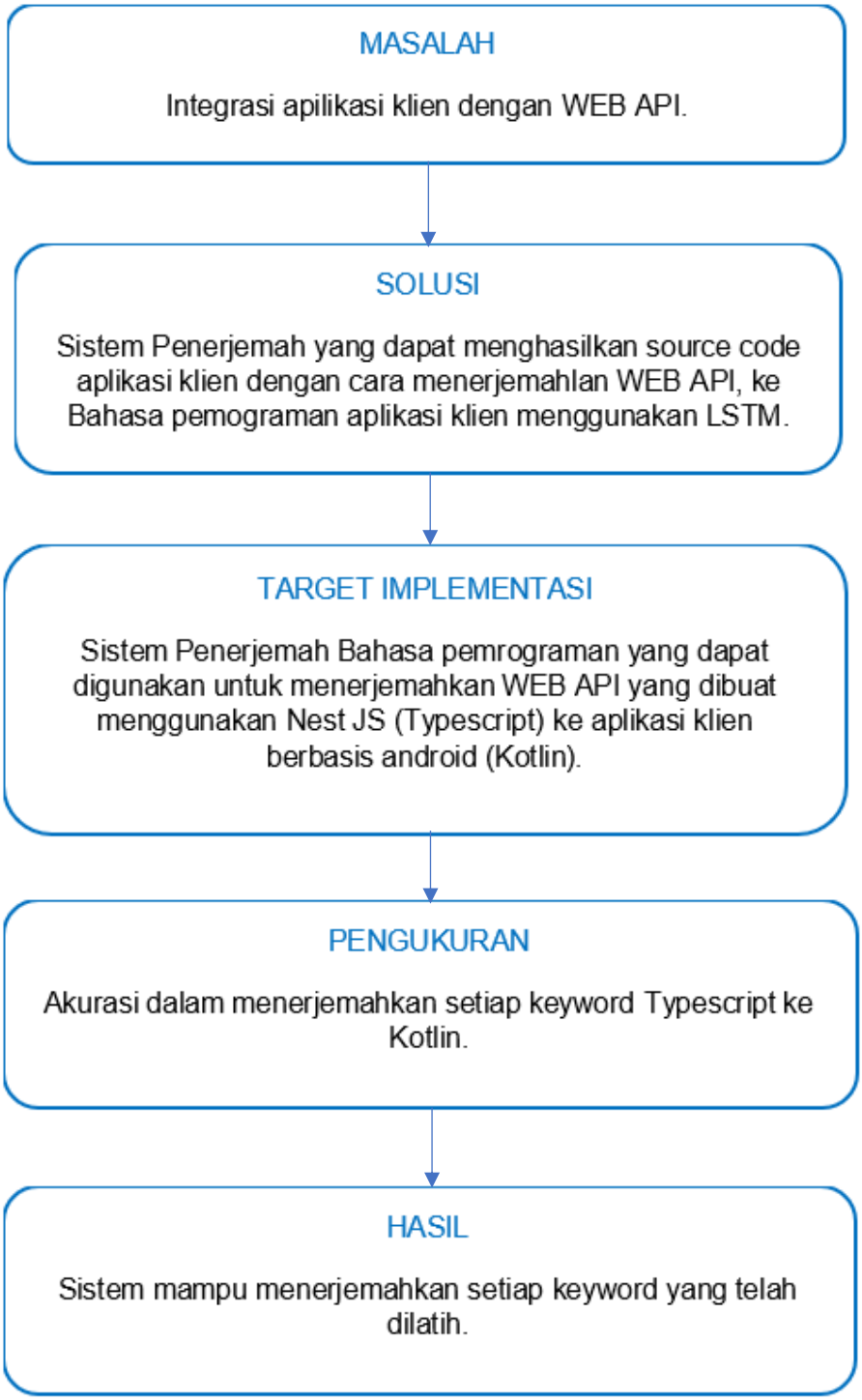
**Tabel 2.2** State Of The Art

| NO | PENULIS/TAHUN | JUDUL | PENERBIT | FITUR | METODE | HASIL |
|---|---|---|---|---|---|---|
| 1 | Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen/2015 | *Divide-and-Conquer Approach for Multi-phase Statistical Migration for Source Code* | *2015 30th IEEE/ACM International Conference on Automated Software Engineering* | *java* ke *c#* | *multi-phase phrase-based SMT (mppSMT)* | migrasi sistem 84.8 - 97.9%, migrasi metode sintaksis dan sematik 70–83%, migrasi metode yang sama persis dengan *c#* 26.3 - 51.2% |
| 2 | Trong Duc Nguyen, Anh Tuan Nguyen, Tien N. Nguyen/2016 | *Mapping API Elements for Code Migration with Vector Representations* | *2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion* | *java api* dalam *jdk* ke *c# api* dalam *.net* | *statistical approach with vector representations* | Akurasi 73.2% |
| 3 | Yusuke Oda, Hiroyuki | *Learning to Generate* | *2015 30th IEEE/ACM* | *python* ke *pseudocode* dalam | *statistical machine* | BLEU *score python* ke |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura /2015 | *Pseudo-code from Source Code using Statistical Machine Translation* | *International Conference on Automated Software Engineering* | bahasa Inggris dan Jepang | *translation (SMT)* | Inggris: 54.08%.<br>BLEU *score python* ke Jepang: 62.88%. |
| 4 | Xinyun Chen, Chang Liu, Dawn Song/2018 | *Tree-to-tree Neural Networks for Program Translation* | *32nd Conference on Neural Information Processing Systems (NIPS 2018), Montreal, Canada* | *java* ke *c#* & *coffescript* ke *javascript*. dan sebaliknya | *tree-to-tree Model* | *tree-to-tree* ungul 75% untuk & *coffescript* ke *javascript* dan sebaliknya dibandingkan dengan *seq2seq, seq2tree, tree2seq*.<br>untuk *java ke c#* dan sebaliknya Peningkatan 20,2%- 41,9%. Dibandingkan *J2C#. lpSMT, mppSMT* |
| 5 | Mehdi Drissi, Olivia Watkins, Aditya Khant, | *Program Language Translation Using a* | *ICML workshop Neural Abstract Machines &* | & *coffescript* ke *javascript* dan | *Grammar-Driven Tree-to-Tree* | Akurasi 88,82% |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Vivaswat Ojha, Pedro Sandoval, Rakia Segev, Eric Weiner, Robert Keller/2018 | *Grammar-Driven Tree-to-Tree Model* | *Program Induction v2 (NAMPI) - Workshop Paper, Stockholm, Sweden, 2018* | sebaliknya. | *Model* | |
| 6 | Ebey Abrahman et al, Akshatha Nayak, and Ashna Iqbal/2019 | *Real-Time Translation of Indian Sign Language using LSTM* | Global Conference for Advancement in Technology (GCAT) Bangalore, India, 2019. | Bahasa isyarat Ke Bahasa India | *LSTM* | Akurasi 98% |
| 7 | Mudiarta Tauda/2021 | penerjemah bahasa pemrograman untuk integrasi aplikasi klien dengan web apis | 2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS) | typescript ke kotlin | *LSTM* | Sistem yang diusulkan dapat menerjemahkan API yang ditulis menggunakan typescript dengan framework nest js ke aplikasi klien berbasis andoid menggunakan libray retrofit untuk koneksi API. |

## D. KERANGKA PIKIR




**Gambar 2.2** Kerangka Pikir