Jurnal Indonesia : Manajemen Informatika dan Komunikasi
www.journal.amikindonesia.ac.id/jimik/ Vol 4 No 2, Mei (2023)
E-ISSN: 2723-7079, P-ISSN: 2776-8074

PEMBANGUNAN *COMPILER DOMAIN SPECIFIC LANGUAGE* SEBAGAI *GENERATOR FORM* HTML MENGGUNAKAN *PYTHON SLY*

Wibisono Adiyoso [1*], Yeremia Alfa Susetyo [2]

[1*,2] Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Kota Salatiga, Provinsi Jawa Tengah, Indonesia.

Email: 672019005@student.uksw.edu [1*], yeremia.alfa@uksw.edu [2]

Histori Artikel:
Dikirim 21 Februari 2023; *Diterima dalam bentuk revisi* 16 Maret 2023; *Diterima* 10 April 2023; *Diterbitkan* 10 Mei 2023.

Abstrak

Perkembangan teknologi informasi yang pesat memberikan dampak signifikan pada cara perusahaan tersebut beroperasi. Perusahaan dapat memanfaatkan teknologi untuk mempermudah dan mempercepat proses bisnisnya, termasuk dalam hal pencatatan dan pengelolaan data transaksi. PT XYZ adalah salah satu perusahaan terbesar di bidang retail, memiliki banyak data dan membutuhkan berbagai jenis form HTML yang berbeda-beda. Oleh karena itu, diperlukan standarisasi dalam penulisan kode program untuk meminimalisir kesalahan dan memudahkan pemeliharaan aplikasi yang dibangun. Penelitian ini bertujuan untuk membangun Compiler Domain Specific Language (DSL) sebagai generator form HTML berbasis web untuk PT XYZ. Penelitian ini menggunakan library Python SLY dengan melewati proses Lexer dan Parser dalam pembangunan DSL. Hasil dari penelitian ini adalah generator form HTML berbasis web yang memudahkan pengembang dalam membuat form HTML secara otomatis dan terstandarisasi. Penggunaan DSL dapat mempercepat proses penulisan kode program, meminimalisir kesalahan, serta menciptakan kode program yang terstruktur dan mudah dipahami oleh para pengembang.

Kata Kunci: Compiler, Domain Specific Language, Form HTML, Python Sly, Konfigurasi, Standarisasi.

Abstract

The rapid development of information technology has a significant impact on the way companies operate. Companies can utilize technology to simplify and accelerate their business processes, including recording and managing transactional data. PT XYZ is one of the largest retail companies, with a vast amount of data and requiring various types of different HTML forms. Therefore, standardization in program code writing is necessary to minimize errors and facilitate application maintenance. This research aims to build a Compiler Domain Specific Language (DSL) as a web-based HTML form generator for PT XYZ. The study uses the Python SLY library, passing the Lexer and Parser processes in DSL development. The result of this research is a web-based HTML form generator that facilitates developers in creating standardized and automatic HTML forms. The use of DSL can speed up the program code writing process, minimize errors, and create structured and easily understood code by developers.

Keyword: Compiler, Domain Specific Language, HTML Form, Python Sly, Configuration, Standardization.

1. Pendahuluan

Saat ini perkembangan teknologi informasi berkembang begitu pesat. Hal tersebut mengubah kebiasaan perusahaan menjadi berorientasi pada teknologi informasi. Banyak perusahaan memanfaatkan perkembangan teknologi untuk mempermudah dan mempercepat proses bisnis. Misalnya untuk melakukan pengelolaan data, perusahaan dapat menggunakan aplikasi berbasis web [1]. Aplikasi berbasis web mampu menghasilkan suatu informasi yang tepat, akurat, dan bermanfaat bagi perusahaan [2].

PT XYZ merupakan salah satu perusahaan terbesar di bidang retail dan memiliki jutaan transaksi yang berlangsung setiap harinya. Pencatatan data merupakan aktivitas yang sangat penting dalam mengelola perusahaan, salah satunya adalah pencatatan data transaksi. Kebutuhan pencatatan data transaksi dapat dilakukan dengan mengisi *form* melalui aplikasi berbasis web [3]. PT XYZ memiliki berbagai jenis data sehingga membutuhkan *form* HTML (*Hyper Text Markup Language*) yang berbeda-beda. Selain itu perusahaan memiliki banyak pengembang yang terbagi dalam berbagai bidang. Hal tersebut dapat menyebabkan ketidakselarasan antara kode program yang ditulis masing-masing pengembang. Sedangkan kode program seharusnya mengutamakan efisiensi dan kemudahan untuk dimengerti oleh pengembang lain [4]. Untuk meminimalisir hal tersebut, dibutuhkan standarisasi agar semua pengembang dapat saling memahami kode program yang ditulis. Standarisasi kode program dapat dilakukan dengan menggunakan *Domain Specific Language*.

Domain Specific Language (DSL) adalah bahasa pemrograman khusus yang didesain untuk menyelesaikan suatu permasalahan spesifik. Namun penggunaan DSL memiliki keterbatasan pada sintaks yang dapat digunakan [5]. Dengan menggunakan DSL, pengembang dapat menentukan kerangka penulisan kode program. Kemudian secara otomatis dapat menghasilkan kode program yang terstandarisasi dan mudah dipahami oleh para pengembang. Selain itu penulisan kode program menjadi lebih efisien karena pengembang tidak harus menulis secara manual. Penggunaan DSL dapat meminimalisir kesalahan penulisan kode program, menciptakan kode program yang terstruktur, serta memudahkan pemeliharaan aplikasi. Sebelum DSL ini terbentuk, harus melewati dua proses yaitu proses Lexer dan juga Parser. Lexer berperan menganalisis kode program untuk membagi sintaks menjadi kata kunci atau token. Token yang sudah dibentuk akan diidentifikasi oleh Parser menjadi susunan instruksi sehingga program dapat dieksekusi [6]. Dari hal-hal yang telah dijabarkan sebelumnya, dapat dirumuskan suatu permasalahan mengenai perancangan standarisasi kode program. Penelitian ini dimulai dengan mengidentifikasi masalah yang ada, melakukan studi literatur, menganalisis metode yang digunakan, hingga pembangunan *Domain Specific Language*. Hasil dari penelitian ini adalah aplikasi generator *form* HTML berbasis web yang dapat digunakan untuk melakukan *generate form* HTML secara otomatis dan terstandarisasi.

Python adalah salah satu bahasa pemrograman tingkat tinggi yang dapat dipakai untuk berbagai tujuan. Bahasa pemrograman Python dapat dijalankan di berbagai platform yang berbeda [7]. Python banyak digunakan oleh pengembang karena Python mudah dipelajari dan bersifat *open-source*. *Compiler* merupakan sebuah program yang digunakan untuk menerjemahkan kode program yang dibuat oleh *programmer* menjadi suatu bahasa mesin [8]. Kode program diubah ke dalam satuan kode biner sehingga kode program tersebut dapat dipahami oleh komputer. *Domain Specific Language* atau DSL adalah suatu bahasa pemrograman khusus yang didesain untuk menyelesaikan masalah yang spesifik dari suatu domain dan mempunyai sintaks yang terbatas. Dengan menggunakan DSL, pengembang dapat menentukan sistem dasar fungsionalitas dan dapat menghasilkan kode program secara otomatis [5].

Python SLY adalah sebuah *library* yang didalamnya menyediakan dua kelas terpisah yaitu Lexer dan Parser. Kelas Lexer digunakan untuk memecah teks masukan menjadi kumpulan token. Token ditentukan dari aturan ekspresi reguler atau *reguler expresion*. Sedangkan, kelas Parser digunakan untuk mengenali sintaks bahasa yang telah digunakan dalam bentuk *file* konfigurasi CFG. *Lexical Analysis* adalah tahap awal dari sebuah *compiler* [9]. Tahapan ini digunakan untuk mengidentifikasi suatu rangkaian dari masukan sumber program suatu bahasa. *Lexical Analyzer* berperan untuk mengkonversi runtun karakter ke dalam runtun kata-kata yang benar, yang dikenal dengan nama Token [10].

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

Python merupakan bahasa pemrograman yang serbaguna dan mudah dipelajari, serta *open-source*. Dapat dijalankan di berbagai platform, sehingga banyak pengembang yang menggunakannya. Compiler merupakan program yang digunakan untuk menerjemahkan kode program menjadi bahasa mesin. Domain Specific Language (DSL) adalah bahasa pemrograman khusus yang dirancang untuk menyelesaikan masalah tertentu dalam suatu domain dengan sintaks yang terbatas. Penggunaan DSL dapat mempercepat proses pembuatan kode program secara otomatis. Python SLY adalah sebuah library yang menyediakan kelas Lexer dan Parser, yang digunakan untuk memecah teks masukan menjadi token dan mengenali sintaks bahasa dalam file konfigurasi CFG. Lexical Analysis adalah tahap awal dari sebuah compiler, dan dilakukan untuk mengidentifikasi rangkaian masukan sumber program dalam suatu bahasa. Lexer berperan dalam mengkonversi karakter ke dalam token, sehingga dapat digunakan oleh Parser untuk memahami sintaks bahasa yang digunakan dalam kode program.

Penelitian berjudul "Analisis dan Perancangan *Domain Specific Language* Untuk Data Generator Pada *Relational Database*" menjelaskan tentang analisis dan perancangan *Domain Specific Language* (DSL) yang dapat digunakan dalam aplikasi data generator [11]. DSL dalam penelitian ini menggunakan masukan berupa *script Data Definition Language* (DDL). Penelitian tersebut menggunakan generator ANTLR yang bertindak sebagai Parser atau *syntax analyzer* untuk mengonversi bahasa menjadi sebuah *grammar*. Sedangkan pada penelitian ini, DSL dikembangkan menggunakan masukan berupa file CFG dan menggunakan Python SLY untuk melakukan konversi DSL. Python SLY memiliki kemampuan *debugging* sehingga mempermudah proses perbaikan dan perubahan sintaks pada kode program. Penelitian lain yang berjudul "Pembangunan *Python Script Generator* Pada Pengembangan Aplikasi Berbasis Web" membahas tentang pembangunan Python Script Generator. Generator tersebut dapat menghasilkan kode program aplikasi berbasis web secara otomatis [12]. Pengguna hanya cukup memilih tabel dari *database* yang akan dibuat menjadi aplikasi berbasis web, lalu sistem akan menghasilkan file berbentuk Python, HTML, dan Yet Another Markup Language (YAML). Pada penelitian tersebut, tidak dijelaskan proses yang terjadi di dalam generator pada saat mengkonversi *string* yang diambil dari *database*. Sedangkan pada penelitian ini akan menjelaskan bagaimana proses dari sebuah *file* konfigurasi berbentuk CFG dikonversi menjadi HTML. Dalam acuan penelitian terakhir yang berjudul "Generator *Form* HTML Berbasis Tabel Dengan Pemrograman Berorientasi Objek" membahas mengenai pembuatan sebuah generator form HTML yang menggunakan suatu tabel *database* sebagai masukan. Penelitian ini menggunakan bahasa pemrograman PHP dan Pemrograman Berorientasi Objek [13]. Pengguna harus memilih tabel dari *database* sehingga generator dapat membaca tag-tag *input*, tipe data, dan panjang data. Pada penelitian tersebut, generator bekerja dengan cara membaca struktur data dari tabel database. Sedangkan pada penelitian ini membaca susunan *string* dalam *file* konfigurasi CFG. Python SLY terintegrasi dengan Python sehingga memudahkan pengembang dalam memahami kode program serta menambahkan fitur baru.

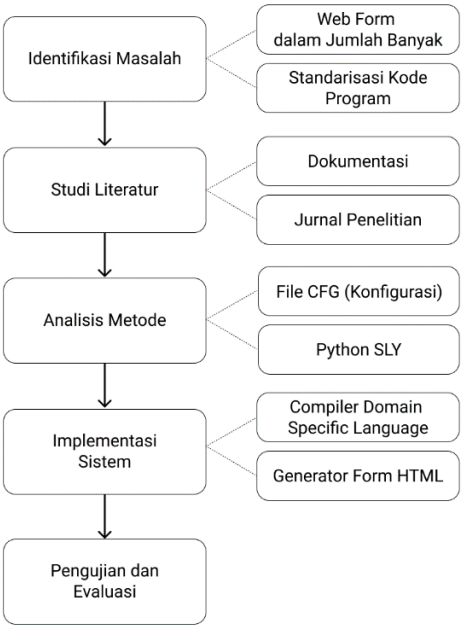
Pembangunan compiler *domain-specific language* (DSL) sebagai *generator* form HTML menggunakan Python SLY memiliki urgensi yang tinggi dalam mengoptimalkan proses pembuatan aplikasi berbasis web yang efisien dan mudah. Penelitian sebelumnya telah membahas pengembangan DSL pada aplikasi data *generator* dan pembuatan *generator* form HTML berbasis tabel menggunakan bahasa pemrograman PHP. Namun, belum ada penelitian yang secara rinci membahas bagaimana sebuah file konfigurasi berbentuk CFG dikonversi menjadi form HTML menggunakan Python SLY. Oleh karena itu, penelitian ini sangat relevan untuk memperdalam pemahaman dan pengembangan DSL yang lebih efisien dan efektif untuk aplikasi berbasis web. Selain itu, dengan menggunakan Python SLY, proses debugging dapat lebih mudah dilakukan dan memungkinkan pengembang untuk menambahkan fitur baru pada aplikasi.

2. Metode Penelitian

Hasil dari penelitian ini adalah Generator *Form* HTML yang dapat digunakan untuk membantu *developer* dalam pengembangan aplikasi web. Penerapan Python SLY dalam pembuatan *Compiler*

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

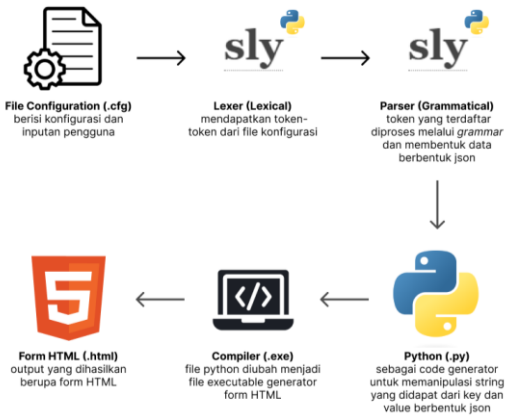
Domain Specific Language secara keseluruhan dilakukan melalui beberapa tahapan yang dapat dilihat pada Gambar 1.



Gambar 1. Metode Penelitian

Pada Gambar 1 menjelaskan setiap proses yang dilakukan dalam penelitian ini, yang pertama adalah mengidentifikasi masalah yang terjadi. Identifikasi masalah dilakukan dengan konsultasi mengenai kebutuhan dan fungsionalitas dari sistem yang akan dibangun. Dari identifikasi masalah tersebut didapatkan informasi bahwa proses pembangunan aplikasi web dengan jumlah *form* yang banyak membutuhkan waktu yang lebih lama. Selain itu, perusahaan membutuhkan standarisasi kode yang digunakan *developer*. Sehingga dibutuhkan sistem yang dapat membantu *developer* dalam proses pembangunan aplikasi web.

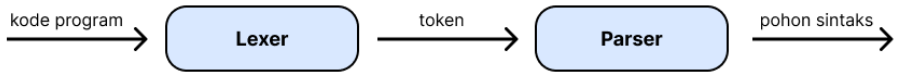
Tahapan selanjutnya setelah identifikasi masalah adalah tahapan studi literatur. Tahapan ini dilakukan dengan mengumpulkan informasi melalui dokumentasi dan jurnal penelitian terdahulu yang berhubungan dengan permasalahan yang diteliti. Tahapan analisis metode dilakukan dengan menentukan metode dan teknologi yang akan digunakan. Dalam pembangunan DSL, Python SLY digunakan sebagai *compiler*. Python SLY memanfaatkan *file* konfigurasi CFG sebagai bahan untuk memproses masukan dan mendapatkan token-token. Kemudian token tersebut akan diolah dan menghasilkan *form* HTML.



Gambar 2. Arsitektur Diagram Sistem

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

Gambar 2 berisikan metode dan teknologi yang digunakan pada penelitian ini. *File* konfigurasi CFG bertindak sebagai masukan dari pengguna dan berisi konfigurasi. Lexer dan Parser merupakan komponen dari Python SLY. Lexer bekerja sebagai *Lexical* untuk mendapatkan token-token dari *file* konfigurasi. Sedangkan Parser bekerja sebagai *Grammatical* untuk memproses token-token melalui *grammar* sehingga menghasilkan susunan sintaks berbentuk *dictionary* Python. Python bertindak sebagai *code generator* untuk memanipulasi *string* dengan mengambil *key* dan *value* yang didapat dari susunan *dictionary*. Maka dihasilkan sebuah bahasa khusus yang sudah terstandarisasi, para pengembang dapat menggunakannya melalui *Compiler Domain Specific Language* untuk melakukan *generate form* HTML. *Form* HTML menggunakan *library* Bootstrap 5. Jika *developer* memiliki kebutuhan *form* yang berbeda, *developer* dapat menggantinya pada bagian *file* konfigurasi CFG.



Gambar 3. Alur Lexer dan Parser

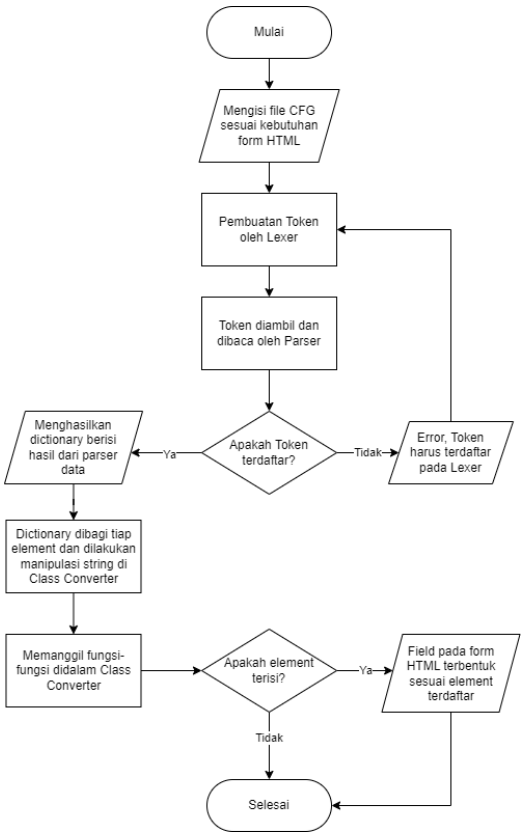
Gambar 3 adalah alur dari proses Lexer dan Parser pada penelitian ini. Pertama, kode program akan dipecah menjadi token oleh Lexer. Token tersebut kemudian digunakan oleh Parser untuk membentuk pohon sintaks. Pohon sintaks atau *syntax tree* adalah sebuah struktur data yang merepresentasikan hubungan antara komponen-komponen pada kode program, seperti operator, variabel, dan fungsi. Pohon sintaks berguna untuk mempermudah proses analisis pada kode program. Selain itu, pohon sintaks membantu dalam menghasilkan kode yang terstruktur dan mudah dipahami.

Setelah tahapan analisis metode, selanjutnya adalah tahapan implementasi sistem. Tahapan implementasi sistem dilakukan dengan mengimplementasikan metode dan teknologi yang digunakan dan merancang aplikasi sesuai dengan kebutuhan dari permasalahan yang ada. Dalam tahapan ini, dilakukan pembangunan *Compiler Domain Specific Language* yang akan bertindak sebagai generator *form* HTML yang sudah terstandarisasi. Tahapan terakhir yaitu melakukan pengujian dan evaluasi. Pengujian yang dilakukan pada penelitian ini adalah *Blackbox testing*. *Blackbox testing* dilakukan dengan pengujian fungsionalitas sistem yang dibangun [14].

3. Hasil dan Pembahasan

Penelitian ini menghasilkan *Compiler Domain Specific Language* sebagai generator *form* HTML yang dapat digunakan *developer* dalam pembangunan sebuah aplikasi web. Generator *form* HTML akan membantu proses pembangunan aplikasi web karena *developer* tidak perlu membuat *form* HTML secara manual. Teknologi yang digunakan untuk membangun generator *form* HTML adalah bahasa pemrograman Python, *library* Python SLY dan file konfigurasi CFG. Generator *form* HTML dapat menghasilkan kode program *frontend* berupa halaman *form* masukan menggunakan *library* Bootstrap. Alur penggunaan dari generator *form* HTML dapat dilihat dari diagram alir pada Gambar 4.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231



Gambar 4. Diagram Alir *Compiler Domain Specific Language*

Pada Gambar 4 dijelaskan alur kerja dari sistem yang telah dibangun. Pertama, pengguna harus mengisi *file* konfigurasi CFG sesuai kebutuhan *form* HTML yang akan digunakan pada aplikasi web. *File* konfigurasi CFG memiliki aturan khusus sehingga pengguna harus mengisi *file tersebut* dengan aturan yang ada. Setelah *file* konfigurasi CFG diisi, sistem akan melakukan proses *execute*. Masukan dari pengguna akan dieksekusi oleh Python SLY melalui proses Lexer untuk mendapatkan token. Kemudian, token-token tersebut akan dibaca dan diolah oleh Parser sehingga menghasilkan suatu struktur. Selama proses *parsing*, terdapat pengecekan token-token, jika token terdaftar pada sistem, maka akan menghasilkan *dictionary* berisi pasangan *key-value* yang akan digunakan sebagai parameter *form* HTML. Tetapi, jika token tidak terdaftar pada sistem, sistem akan mengembalikkan *error* kepada pengguna, sehingga pengguna perlu untuk merubah masukan atau mendaftarkan token pada sistem. *Key* adalah *identifier* atau nama yang digunakan untuk memberikan identitas untuk setiap elemen. Sedangkan *Value* adalah nilai dari sebuah elemen tersebut. *Form* HTML memiliki beberapa elemen yang dapat digunakan. *Dictionary* yang berisi pasangan *key-value* terbagi kedalam tiap-tiap elemen *form* HTML dan sistem akan melakukan proses manipulasi *string*. Hasil manipulasi *string* akan digunakan pada setiap fungsi sehingga *form* HTML yang terbentuk sesuai dengan elemennya masing-masing. Jika elemen tidak terisi, maka kolom *inputan* tidak akan terbentuk. Selanjutnya jika elemen terisi, maka *field* atau kolom masukan *form* HTML akan terbentuk pada *form* aplikasi web.

Kode Program 1. Potongan *File* Konfigurasi CFG

```
TITLE: Kerja Praktek
HEADER: Form Data Mahasiswa Kerja Praktek
TEXTBOX:
    CAPTION: Nama Lengkap
    NAME: namalengkap
    ID: nama
    LENGTH: 100
```


Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

```
    TYPE: text
…
RADIOBUTTON:
    CAPTION: Jenis Kelamin
    NAME: jeniskelamin
    …
    OPTION:
        OPTVALUE: Wanita
        OPTID: wanita
```

Kode Program 1 merupakan potongan dari *file* konfigurasi CFG. Pengguna dapat melakukan konfigurasi pada *file* sebagai masukan dari pengguna. *File* CFG terdiri dari 2 bagian, yaitu elemen dan juga sub elemen. Elemen adalah *field* yang dapat digunakan sebagai *inputan form* HTML seperti *input text*, *number, email, password, radio, checkbox,* dan lain-lain. Sedangkan sub elemen digunakan sebagai parameter yang akan digunakan pada *form* HTML. Konfigurasi CFG digunakan untuk melakukan proses Lexer dengan memecah masukan menjadi beberapa token. Proses ini disebut sebagai proses *lexing*.

Kode Program 2. Kelas *startLexer*

```
from sly import Lexer
class startLexer(Lexer):
    tokens = {
        TEXTBOX,
        ...
        COLON,
        DATA,
    }
    …
    TEXTBOX         = r"(?i)TEXTBOX"
    …
    COLON           = r"\:"
    DATA            = r'[a-zA-Z0-9][a-zA-Z0-9 ]*'
```

Kode Program 2 adalah kelas *startLexer* yang berisi kumpulan token yang didaftarkan pada sistem penelitian ini. Kelas ini menggunakan *library* SLY dengan mengimport fungsi yang bernama Lexer. Dalam Python SLY, aturan ekpresi reguler atau *regex* didefinisikan sebagai token. *Regex* adalah sebuah pola teks yang mendefinisikan pola pencarian untuk suatu teks [15]. Token dapat didaftarkan pada bagian tokens sehingga sistem dapat mengenali setiap masukan dari pengguna atau *developer*. Proses *Lexing* akan mencocokan *string* masukan dengan aturan *regex* yang sudah didaftarkan. Jika *string* masukan cocok dengan aturan *regex*, Lexer akan menghasilkan token. Tetapi jika tidak ada aturan *regex* yang cocok dengan *string* masukan, Lexer akan membuat pesan kesalahan dan tidak menerima token. Setelah semua *string* masukan dari file CFG diproses, Lexer akan mengembalikan sekumpulan token kepada Parser. Kemudian akan dilanjutkan ke dalam proses yang bernama *Parsing*.

Kode Program 3. Kelas startParser

```
class startParser(Parser):
    tokens = startLexer.tokens
    @_("")
    def configuration(self, p):
        return {}
    @_("mainconfig configuration")
    def configuration(self, p):
        dictconfiguration = p.configuration
        key, subconfig = p.mainconfig
        if key in dictconfiguration:
            dictconfiguration[key].append(subconfig)
        …
    @_("TEXTBOX COLON childconfig")
    def mainconfig(self, p):
```


Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

```
        return 'TEXTBOX', p.childconfig
    ...
    @_("DATA")
    def expr(self, p):
        return p.DATA
```

Pada Kode Program 3 merupakan kelas *startParser* untuk memulai proses *Parsing*. Variabel tokens adalah *list* token yang di-*generate* oleh Lexer pada kelas *startLexer*. Fungsi *startParser* mengubah *string* masukan menjadi sebuah pohon sintaks. Proses *Parsing* membagi masukan menjadi token-token dan kemudian menganalisis hubungan antar token-token tersebut. Parser memiliki beberapa *method* yang digunakan seperti *configuration, mainconfig, childconfig,* dan *expr*. *Method* digunakan untuk produksi grammar dalam membangun pohon sintaks. Masing-masing *method* memproses masukan dan menghasilkan suatu keluaran berupa *dictionary* atau *string*. Setiap *method* memiliki *decorator* "@_" yang berfungsi sebagai aturan *grammar* pada masukan. Hasil atau *output* dari proses *Parsing* berupa *dictionary* atau *string* yang menyimpan informasi konfigurasi dan data.

```
WARNING: Token {TEXTBOX} defined, but not used
WARNING: There is 1 unused token
sly: Syntax error at line 3, token=TEXTBOX
```

Gambar 5. Pesan Kesalahan Pada Saat Parsing

Gambar 5. merupakan gambar pesan kesalahan yang dihasilkan ketika proses *Parsing* berjalan. Pesan kesalahan adalah salah satu fitur yang diberikan oleh Python SLY untuk membantu *developer* mengetahui *error* yang terjadi dalam pembangunan DSL. Pesan kesalahan tersebut terjadi ketika token "TEXTBOX" terdaftar akan tetapi pada saat proses *Parsing* tidak ada *grammar yang* mengatur token "TEXTBOX". *Developer* dapat menambahkan *grammar* aturan sesuai kebutuhan sistem sehingga proses *Parsing* dapat dilanjutkan.

Kode Progam 4. Kelas Converter dan fungsi elemen Textbox

```
class Converter:
    …
def setElementTextbox(self, htmlScript):
        for element, elementConfig in self.dict_config.items():
            …
            script += str(subElement).lower() + '="' + str(subElementConfig[0])
+ '" '
                    if subElement == elementName:
                        break
            self.generateEndDivHTML(htmlScript)
            htmlScript.append(script + ">")
            self.generateStartDivHTML(htmlScript, forValue, captionValue)
```

Kode Program 4 adalah dimulainya proses manipulasi *string* setelah proses Parser selesai dijalankan. Proses Parser menghasilkan *dictionary* yang berbentuk seperti Json. Hasil *dictionary* berkebalikan dengan masukan *developer* pada file konfigurasi CFG karena proses Parser membaca *string* dan sintaks dari bawah. Hal ini tidak menjadi masalah karena di Python mempunyai fungsi bernama Reverse(). Fungsi Reverse() berfungsi untuk mengembalikan urutan elemen pada *dictionary*.

Kode Program 4 merupakan definisi kelas Converter dan fungsi setElementTextbox. Fungsi setElementTextbox mempunyai satu parameter bernama htmlScript yang merupakan sebuah *list*. Fungsi ini menggunakan perulangan *for* untuk memproses data di dalam sebuah *dictionary*. Terdapat variabel element dan elementConfig yang digunakan untuk menyimpan setiap pasangan *key-value* yang ada dalam sebuah *dictionary* self.dict_config. Fungsi dari self.dict_config.items() adalah mengambil setiap pasangan *key* dan *value* dalam *dictionary* self.dict_config. Kemudian pasangan *key-value* yang telah diambil dan disimpan nilainya dalam sebuah *tuple*. *Tuple* akan diteruskan pada perulangan *for*

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

selanjutnya. Dengan demikian kode program dapat mengakses setiap elemen yang ada dalam *dictionary* self.dict_config.

Dictionary yang sudah didapat setelah proses Parser, dilakukan proses manipulasi *string* sehingga didapatkan kunci dan nilai-nilai yang digunakan dalam pembuatan generator *form* HTML. Tujuan dari kode program ini adalah untuk membangun sebuah tag *input* dalam HTML yang sesuai dengan konfigurasi yang diberikan dan menambahkannya ke dalam *array* htmlScript. Tag *input* tersebut akan digunakan untuk menampilkan elemen yang sesuai pada halaman web yang dihasilkan.

Kode Program 5. Fungsi elemen Button

```
def setElementButton(self, htmlScript):
   for element, elementConfig in self.dict_config.items():
      if(element == "BUTTON"):
         for termConfig in elementConfig:
            …
            script += str(subElement).lower() + '="' + str(subElementConfig[0])
+ '" '
            self.generateEndDivHTML(htmlScript)
            htmlScript.append('                </div>')
            htmlScript.append(script + "><br>")
            htmlScript.append('<div  class="d-grid  gap-2  d-md-flex  justify-
content-md-center">')
            self.generateStartDivOriginalHTML(htmlScript)
```

Kode Program 5 merupakan lanjutan dari Kode Program 4. Kode Program 5 adalah bagian dari kelas Converter yang memiliki fungsi untuk menambahkan elemen Button pada sebuah skrip HTML. Pada fungsi setElementButton(), dilakukan perulangan *dictionary* dict_config yang memiliki *key* "BUTTON". ElementConfig merupakan *dictionary* yang berisikan konfigurasi dari elemen BUTTON yang didapatkan setelah proses Parsing *file* konfigurasi CFG. Setiap elementConfig akan diproses untuk membuat skrip HTML untuk masing-masing elemen yang tersedia.

Kode Program 6. Fungsi generate semua elemen

```
def generateAllElement(self, htmlScript, headerValue, titleValue):
        self.generateEndBodyHTML(htmlScript)
        for element, elementConfig in self.dict_config.items():
            if element == "TEXTBOX":
                self.setElementTextbox(htmlScript)
            elif element == "DATETIME":
                self.setElementDatetime(htmlScript)
            …
        self.generateStartBodyHTML(htmlScript, headerValue)
        self.generateHeadHTML(htmlScript, titleValue)
```

Setelah semua elemen pada *dictionary* berhasil dipecah dan dilakukan manipulasi *string*, maka akan dilakukan pemanggilan fungsi selanjutnya. Fungsi generateAllElement adalah fungsi yang digunakan untuk memanggil beberapa fungsi lain seperti fungsi setElementTextbox, fungsi setElementDatetime, fungsi setElementCheckbox, dan fungsi-fungsi lain yang memproses elemen *form* HTML. Fungsi ini menerima tiga parameter yaitu htmlScript, headerValue, serta titleValue. Parameter htmlScript digunakan untuk menyimpan hasil *output* HTML. Sementara parameter headerValue dan titleValue digunakan untuk memasukkan *header* dan *title* pada halaman HTML.

Dalam fungsi generateAllElement, nilai elemen yang didapatkan dari self.dict_config diperiksa satu persatu dengan perulangan *for* dan pengkondisian *if* dan *elif*. Jika terdapat elemen "TEXTBOX", maka fungsi setElementTextbox akan dipanggil. Begitu juga dengan elemen yang lainnya, jika elemen ditemukan, maka setiap fungsi elemen tersebut akan dipanggil.

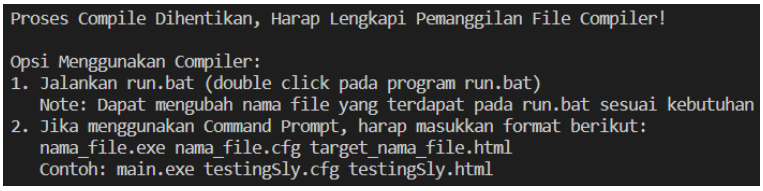
Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

Kode Progam 7. Fungsi utama *Compiler* Domain Specific Language

```
def main():
    …
        with open(fname, 'w') as f:
    …
if __name__ == '__main__':
    …
            with open(filename, 'r') as file:
                data = """"""
                for i in file:
                    data += i
                lexer = startLexer()
                parser = startParser()
                dictConfig = parser.parse(lexer.tokenize(data))
                file.close()
                converter = Converter(dictConfig)
                converter.generateAllElement(htmlScript,          headerValue,
titleValue)
    …
    htmlScript.reverse()
    main()
```

Kode Program 7 merupakan bagian utama dari pembangunan *Compiler Domain Specific Language* yang menggunakan *library* Python SLY untuk melakukan proses *Lexing* dan *Parsing*. Kode program "with open(fname, 'w') as f: ... f.close()" akan membuka *file* dengan nama "fname" akan menulis setiap baris hasil konversi yang ada di dalam *array* htmlScript. Fungsi main akan digunakan pada akhir program.

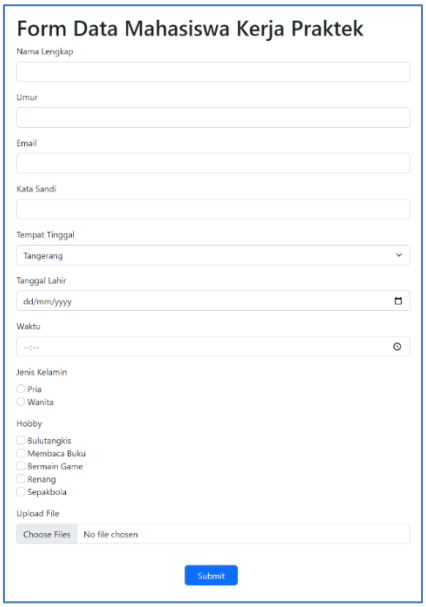
Selanjutnya, program akan mengecek pada bagian argumennya, jika argumen pada *command line* lebih dari 2, maka sistem akan melakukan proses *generate file form* HTML. Akan tetapi, jika argumen tidak lebih dari 2, maka akan keluar pesan kesalahan sehingga *developer* harus melakukan perbaikan perintah pada *command line*. Pada fungsi utama ini, melakukan pemanggilan *dictionary* dan melakukan penulisan ke dalam *file form* yang akan di *generate*. Sebelum itu, *array* htmlScript dilakukan pemanggilan fungsi *Reverse*() untuk mengembalikan urutan script *form* HTML dan fungsi main akan dipanggil. Setelah semua fungsi berhasil dijalankan, kode program Python di konversi ke dalam *file executable* (.exe) sehingga pengguna hanya memerlukan satu *file* untuk dieksekusi dan melakukan *generate form* HTML.



Gambar 6. Pesan Kesalahan Pada Generator

Gambar 6 adalah pesan kesalahan yang akan muncul ketika *developer* salah atau kurang memasukkan argumen yang dibutuhkan oleh sistem. Pesan kesalahan tersebut dibuat agar *developer* dapat menggunakan *Compiler* DSL dengan benar dan juga dapat melakukan proses *generate file form* HTML untuk pembangunan aplikasi berbasis web.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231



Gambar 7. *Form* HTML Hasil dari Generator

Gambar 7 adalah tampilan *form* HTML hasil dari generator. *Field* masukan yang ada di *form* HTML sesuai dengan masukan pengembang pada *file* konfigurasi CFG. Jika kebutuhan *form* berbeda, pengembang dapat mengaturnya pada bagian *file* konfigurasi CFG dengan menghapus bagian yang tidak dibutuhkan. Setelah *form* berhasil di *generate*, pengembang dapat melanjutkan pembangunan aplikasi *form* berbasis web dengan kode program yang sudah terstandarisasi. Langkah terakhir adalah melakukan pengujian dari Generator yang telah dibangun. Pengujian sistem dilakukan menggunakan *Blackbox testing*. Pengujian menggunakan *Blackbox testing* dilakukan untuk mengetahui apakah hasil penelitian dapat berfungsi dengan baik atau tidak. Hasil dari pengujian Generator menggunakan *Blackbox testing* dapat dilihat pada Tabel 1.

Tabel 1. Hasil pengujian *Blackbox testing*

Pengujian	Hasil yang diharapkan	Hasil Pengujian	Status Pengujian
Generate file form HTML dengan *file executable* (.exe) dengan dua argumen yaitu *file CFG* dan tujuan *file form* HTML.	Generator akan membentuk *file form* HTML sesuai dengan masukan yang ada pada *file* konfigurasi CFG.	*File form* HTML terbentuk sesuai dengan masukan yang ada pada *file* konfigurasi CFG.	Valid
Generate file form HTML dengan *file executable* (.exe) tanpa argumen apapun	*File form* HTML tidak terbentuk dan keluar pesan kesalahan dan informasi yang sudah diatur sehingga pengembang dapat melakukan perbaikan pada saat melakukan *generate file*.	Akan keluar pesan kesalahan dan informasi yang sudah diatur sehingga pengembang dapat melakukan perbaikan pada saat melakukan *generate file* dan *file form* HTML tidak terbentuk.	Valid
Memasukan *string* elemen pada *file* konfigurasi CFG yang tidak terdaftar pada Token	Akan terjadi kesalahan pada program dan keluar pesan kesalahan.	Terjadi kesalahan pada program dan menampilkan pesan kesalahan yang dihasilkan dari *library* Python SLY.	Valid

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 4 No 2, Mei (2023), pp. 449-461
DOI : https://doi.org/10.35870/jimik.v4i2.231

4. Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa pembangunan sebuah *Compiler Domain Specific Language* sebagai generator *form* HTML menggunakan Python SLY dengan *file* konfigurasi CFG berhasil dilakukan. Python SLY memberikan kemudahan dalam pembuatan Lexer dan Parser untuk bahasa DSL yang dibuat. Pemilihan Python SLY sebagai *library* dalam penelitian ini karena Python SLY memiliki dokumentasi yang baik dan mudah dipahami, sehingga memudahkan *developer* dalam mengembangkan dan melakukan pemeliharaan sistem. Penelitian ini menghasilkan sebuah bahasa DSL yang bekerja sebagai generator *form* HTML dan membantu *developer* dalam pembuatan *form* HTML secara otomatis. Generator *form* HTML yang dihasilkan akan digunakan PT. XYZ dalam pengembangan aplikasi web yang membutuhkan banyak *form* HTML. Dari hasil penelitian yang dilakukan, Python SLY dapat menampilkan pesan kesalahan yang terperinci dan memberikan petunjuk yang jelas tentang penyebab kesalahan. Pesan kesalahan pada Python SLY memberikan informasi seperti baris dan kolom letak kesalahan terjadi, jenis kesalahan, serta deskripsi mengenai kesalahan tersebut. Dengan demikian, *developer* dapat menemukan dan memperbaiki kesalahan sintaksis pada kode program.

Sebagai saran untuk pengembangan selanjutnya, penulis menyarankan untuk melakukan perluasan kemampuan *Compiler* DSL dengan menambahkan fitur-fitur tambahan, seperti pengolahan data dan validasi *input form*. Selain itu, dapat juga dilakukan pengembangan dengan mengimplementasikan teknologi lain, seperti PyParsing atau ANTLR untuk membandingkan kelebihan dan kekurangan masing-masing dalam pembuatan *Compiler Domain Specific Language*.

5. Daftar Pustaka

[1] Nursyanti, R., Alamsyah, R. Y. R. and Perdana, S., 2019. Perancangan Aplikasi Berbasis Web Untuk Membantu Pengujian Kualitas Kain Tekstil Otomotif. *Explore – Jurnal Sistem Informasi dan Telematika*, *10*(2), pp. 153-159.

[2] Tarigan, R. and Ardiansyah, D., 2022. Perancangan Aplikasi Inventory Barang pada CV. MR Lestari Berbasis Web. *Jurnal SIMIKA*, *3*(2), pp. 77-94. DOI: https://doi.org/10.47080/simika.v3i2.985.

[3] Endra, R. Y., Aprilinda, Y., Dharmawan, Y. Y. and Ramadhan, W., 2021. Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website. *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, *11*(1), pp. 48-55. DOI: http://dx.doi.org/10.36448/expert.v11i1.2012.

[4] Ningtyas, D. F. and Setiyawati, N., 2021. Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request. *Jurnal Janitra Informatika dan Sistem Informasi*, *1*(1), pp. 19-34. DOI: https://doi.org/10.25008/janitra.v1i1.120.

[5] Halomoan, M. A., Kharisma, A. P. and Marji, 2021. Pengembangan Domain Specific Language Untuk Aplikasi CRUD Berbasis Web. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *5*(1), pp. 34-41. DOI: https://doi.org/10.37792/jukanti.v6i1.916.

[6] Jonathan, B., Avetyan, R. and Abeln, S., 2019. Creating Domain-Specific Language and Syntax Checker Using Xtext. *JIRAE*, *4*(1), pp. 26-32.

[7] Lisia, V., Widjajam, A. E., Mitra, A. R., Haryani, C. A. and Hery, 2022. Visualisasi Data Bencana Geologi di Indonesia Berbasis Web. *Information System Development (ISD)*, *7*(1), pp. 9-27. DOI: https://doi.org/10.9744/jirae.4.1.26-32.

[8] Soetiman, R. V., Poekoel, V. C. and Kambey, F. D., 2021. Song Search Application Using Fast Fourier Transform Method. *Jurnal Teknik Elektro dan Komputer*, *10*(1), pp. 27-36.

[9] Alfred, V. A., Monica S. Lam, Ravi Sethi, dkk., *Compilers : Principles, Techniques, & Tools (Second Edition)*. United States of America: Pearson Education, Inc.

[10] Harahap, M., 2022. Perancangan Perangkat Lunak Teks Editor Bahasa C Menggunakan Metode Lexical Analyzer. *Buletin Ilmiah Informatika Teknologi (BIIT)*, *1*(1), pp. 8-14.

[11] Natali, V. and Alfadian, P., 2019. Analisis dan Perancangan Domain Specific Language untuk Data Generator pada Relational Database. *Jurnal Masyarakat Informatika Unjani (JUMANJI)*, *3*(1), pp. 64-73. DOI: https://doi.org/10.26874/jumanji.v3i01.52.

[12] Wicaksono, H. A. and Setiyawati, N., 2022. Pembangunan Python Script Generator pada Pengembangan Aplikasi Berbasis Web. *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, *5*(1), pp. 157-166. DOI: https://doi.org/10.37792/jukanti.v5i1.472.

[13] Wijana, K., 2019. Generator Form HTML Berbasis Tabel Dengan Pemrograman Berorientasi Objek. *JUTEI (Jurnal Terapan Teknologi Informasi)*, *3*(1), pp. 11-20. DOI: https://doi.org/10.21460/jutei.2019.31.147.

[14] Gumilar, M. D., Sembiring, F. and Erfina, A., 2021. Implementasi Progressive Wehb App Pada Sistem Informasi E-Learning Untuk Pembelajaran Bahasa Pemgrograman Python. *JUTISI: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi*, *10*(2), pp. 309-318. DOI : https://doi.org/10.35889/jutisi.v10i2.658.

[15] Safitri, S. I., Suhery, C. and Bahri, S., 2021. Implementasi Algoritma K-Means Untuk Clustering Sentimen Pada Opini Kualitas Pelayanan Jasa Penerbangan. *Coding: Jurnal Komputer dan Aplikasi*, *9*(2), pp. 186-197. DOI: http://dx.doi.org/10.26418/coding.v9i02.47377.