

PROYECTO INTEGRADOR

TEORÍA DE LA COMPUTACIÓN

LOZA SANDOVAL LEONARDO SEBASTIAN
222790633

ARELLANO GRANADOS ANGEL MARIANO
218123444

18/05/2022

Validador/generador de correos electrónicos institucionales para alumnos de la UDG

Planteamiento del problema

Todos los alumnos y maestros de la UDG contamos con un correo institucional dado por el SIIAU, este nos reconoce como parte de la universidad y nos ofrece muchas ventajas por encima de una de una cuenta de correo normal; todos estos correos son generados con un mismo formato para los alumnos, en este proyecto se plantea la creación de una herramienta que valide y genere correos institucionales con algunos datos de los alumnos, para así agilizar el proceso de creación con ayuda de un autómata de pila.

Objetivos

El objetivo de este proyecto es desarrollar un autómata de pila que permita validar direcciones de correo electrónico institucionales de acuerdo con los estándares establecidos. Esto implica verificar que la dirección cumpla con la estructura requerida, como la presencia de un nombre de usuario, seguido de un símbolo "@" y un dominio válido. Además, se pueden implementar reglas adicionales para garantizar que las direcciones sean exclusivas de una institución específica. En este caso, hablamos del nombre de dominio de la red universitaria de UDG para alumnos.

También se contempla generar correos electrónicos válidos, en donde se le solicita al usuario nombre y apellido y se genera un correo electrónico con el formato especificado.

El formato exacto buscado es: nombre del alumno, punto, apellido del alumno, 4 números aleatorios, arroba '@' y la terminación "alumnos.udg.mx".

Justificación

La validación de direcciones de correo electrónico institucionales es un problema importante en la comunicación institucional y puede tener un impacto significativo en la seguridad y eficiencia de la comunicación. A menudo, las direcciones de correo electrónico pueden contener errores tipográficos, caracteres inválidos o pueden no cumplir con los estándares establecidos, lo que puede llevar a problemas en la entrega de mensajes y la pérdida de información valiosa.

En este sentido, la teoría de la computación y los autómatas de pila pueden ofrecer un enfoque riguroso y sólido para abordar este problema. La utilización de un autómata de pila para la validación de direcciones de correo electrónico

institucionales puede garantizar la precisión y eficiencia de la verificación del formato de las direcciones de correo electrónico.

Además, el desarrollo de un autómata de pila para la validación de direcciones de correo electrónico institucionales puede tener una amplia gama de aplicaciones en el ámbito institucional y empresarial. Por ejemplo, puede utilizarse en la validación de direcciones de correo electrónico de clientes o proveedores, en el registro de usuarios en sistemas internos, en la gestión de bases de datos de contactos y en el monitoreo de la comunicación interna.

Marco teórico

En el ámbito de la teoría de la computación, los autómatas de pila son una clase de máquinas abstractas que desempeñan un papel fundamental en el análisis y validación de lenguajes formales. Estos autómatas utilizan una estructura de datos llamada pila para almacenar información mientras procesan cadenas de entrada.

La validación de direcciones de correo electrónico institucionales se refiere a la verificación de que una dirección de correo electrónico cumpla con el formato requerido y esté de acuerdo con los estándares establecidos. Esto implica asegurarse de que la dirección contenga una serie de elementos esenciales, como un nombre de usuario, seguido de un símbolo "@" y un dominio válido.

Al aplicar la teoría de la computación y los autómatas de pila a la validación de correos electrónicos institucionales, se puede desarrollar un autómata de pila específico para este propósito. Este autómata se configuraría con un conjunto de reglas y transiciones que permitan verificar de manera eficiente y precisa el formato y la estructura de las direcciones de correo electrónico.

El autómata de pila utilizado en este proyecto tendría un conjunto de estados que representan diferentes etapas del proceso de validación. Cada estado estaría asociado con una serie de transiciones que determinan cómo la máquina cambia de estado en función de la entrada y del contenido actual de la pila.

Además, el autómata de pila puede aplicar reglas adicionales para validar elementos específicos de las direcciones de correo electrónico institucionales, como la exclusividad de un dominio para una institución en particular o restricciones adicionales en el formato del nombre de usuario.

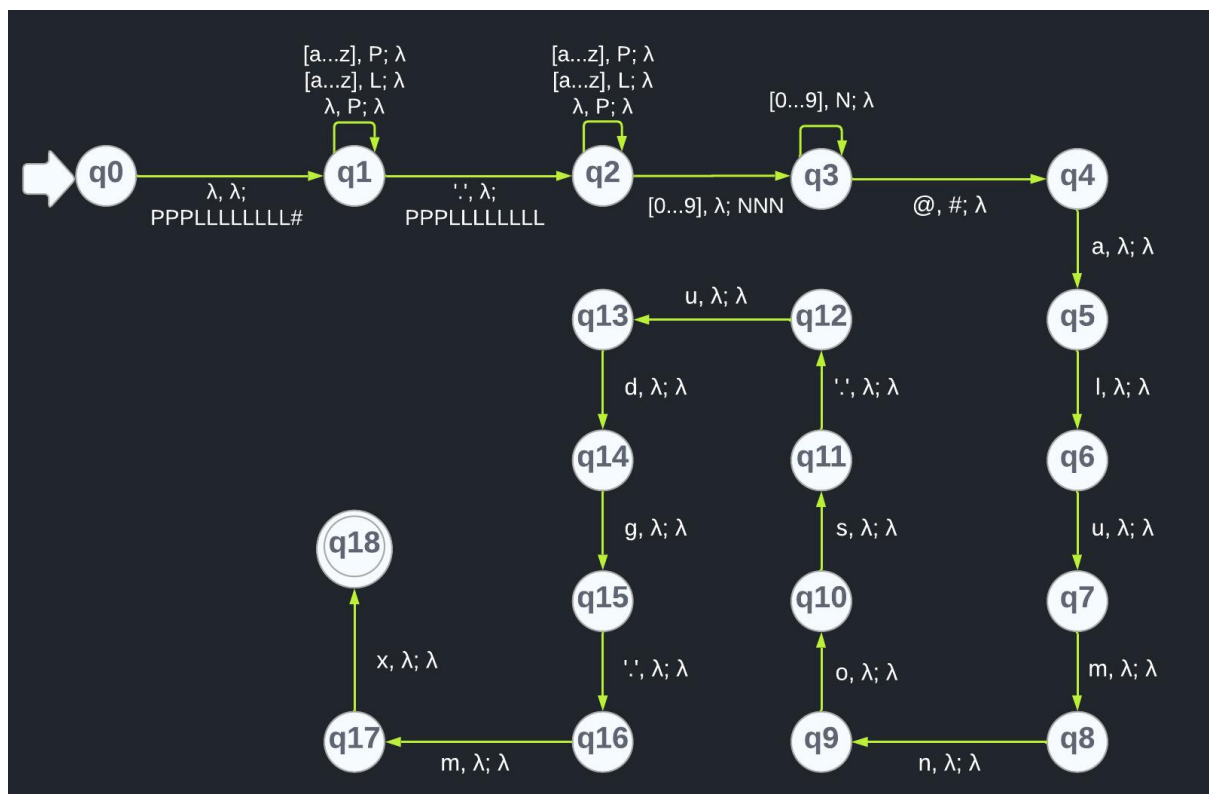
La utilización de un autómata de pila para validar correos electrónicos institucionales ofrece varias ventajas. En primer lugar, permite realizar una verificación exhaustiva y precisa del formato de las direcciones de correo electrónico, evitando errores comunes y garantizando la integridad de la comunicación institucional. Además, al

estar basado en la teoría de la computación, proporciona un enfoque sólido y riguroso para abordar este desafío.

En resumen, este proyecto se basa en la teoría de la computación y los autómatas de pila para desarrollar un enfoque eficiente y preciso para la validación de correos electrónicos institucionales. El autómata de pila diseñado permitirá verificar el formato y la estructura de las direcciones de correo electrónico, asegurando la calidad y confiabilidad de la comunicación institucional.

Metodología

Primero se creó el diagrama de estados en papel pasando por varias versiones hasta que nos decidimos por utilizar el autómata de pila, era necesaria la implementación de la pila porque así tendríamos un control sobre la cantidad mínima y máxima de caracteres tanto en el nombre y el apellido, también nos ayudaría a asegurarnos de que solo tendría cuatro números después del apellido.



Diseño final del diagrama

La manera en la que funciona la pila es que en el estado anterior donde se pretende que se reciban los caracteres del nombre y apellido se apilan tres letras 'P', ocho letras 'L' (o menos dependiendo el máximo de caracteres) y solo en la primera inserción un carácter '#' para marcar el final de la cadena; con la pila llena de estos

caracteres sólo se podrá desapilar si se ingresan letras minúsculas con mínimo tres letras desapilado las 'P' y un máximo 11 letras desafiando todas las 'L', el nombre no necesita desapilar todas las 'L' hay una transición lambda que desapilar 'L' sin que entre ningún carácter, este mismo funcionamiento se aplica a los números donde al momento que se ingresa un número cambia de estado apilando tres 'N', que se desapila ingresando otros tres números, así teniendo los cuatro y completando el formato, por último todos los correos terminan con la misma cadena que no varía después de los números, me refiero a '@alumnos.udg.mx' donde se necesita un estado en cada uno de los caracteres, pues un error en cualquiera de estos caracteres invalidaría la cadena, el único estado especial es en donde se recibe la '@' pues se usa como bandera para asegurarnos que la cadena anterior salió bien, pues esta desapila el '#apilado en el primer estado.

Ya hablando de la implementación se seleccionó el lenguaje python y se creó una interfaz gráfica para contener las dos funciones del programa validar cadenas y crear cadenas.

En cuanto al código se codificó en Visual Studio Code contando con las clases Matriz para guardar las tradiciones de cada estado, la clase Pila para lograr implementar la pila en el programa y la clase MainWindow que se conecta a la interfaz gráfica (ui_mainWindow.py) para asignar funciones y comportamientos a los botones y líneas de texto, de donde se desprende nuestras dos funciones principales validar y crear contenidas en las pecanas con los mismos nombres.

La primera pestaña corresponde a la validación de correos, que solicita escribir la cadena a validar en un recuadro y presionar un botón para comenzar el proceso.

MainWindow

Validador y creador de cadenas en el formato de correos institucionales de UDG

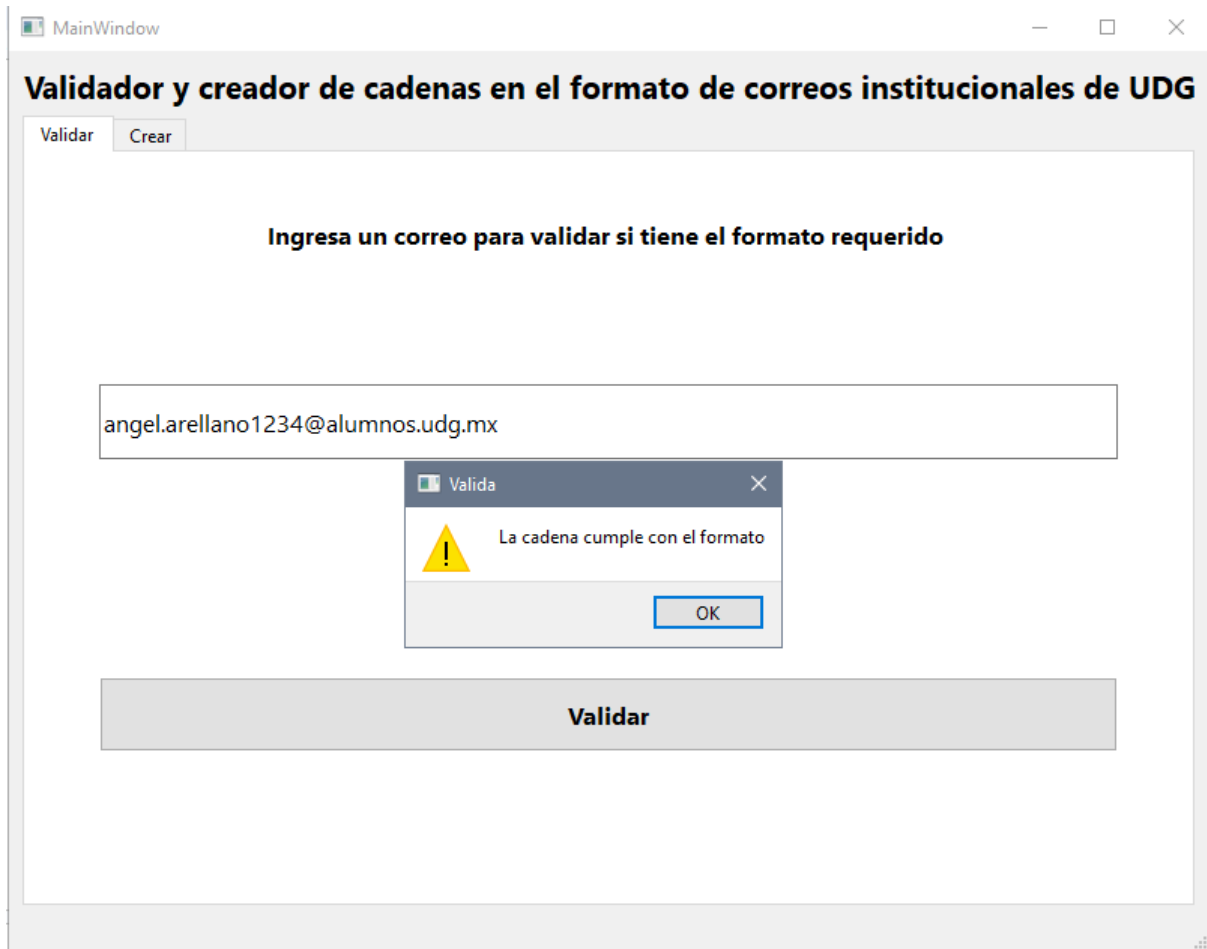
Validar Crear

Ingresa un correo para validar si tiene el formato requerido

Ingresa un correo de la UDG

Validar

Si insertamos una cadena correcta cómo sería mi correo institucional personal “angel.arellano1234@alumnos.udg.mx” podremos observar una ventana emergente que nos confirma que la cadena cumple con el formato.



También podemos observar el proceso que sigue el algoritmo para validar cada carácter, así como el contenido de la pila dentro de la consola de nuestro IDE.

```

PS J:\Trabajos 5 CUCEI\5. Teoria de la Computacion\proyecto\codigo> & C:/Users/Servidor/AppData/Local/Programs/Python/Python39-64/Python.exe .py"
pila Inicial: ['P', 'P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']

Entra: a
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['P', 'P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: P

Entra: n
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: P

Entra: g
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: P

Entra: e
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: L

Entra: l
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: L

Entra: .
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPP', 2)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
POP: L
POP: L
POP: L
POP: L
POP: L
POP: L
POP: L
pila: ['P', 'P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']

Entra: a
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['P', 'P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: P

Entra: r
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: P

```



```
Entra: e
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: P

Entra: l
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L

Entra: l
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L

Entra: a
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L

Entra: n
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L

Entra: o
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L

Entra: 1
estado q2:[('letMin', 'P', 'lam', 2), ('letMin', 'L', 'lam', 2), ('lam', 'L', 'lam', 2), ('lam', 'lam', 'NNN', 3)]
Pila: ['L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']2
POP: L
POP: L
POP: L
pila: ['N', 'N', 'N', '#']

Entra: 2
estado q3:[('num', 'N', 'lam', 3), ('@', '#', 'lam', 4)]
Pila: ['N', 'N', 'N', '#']3
POP: N

Entra: 3
estado q3:[('num', 'N', 'lam', 3), ('@', '#', 'lam', 4)]
Pila: ['N', 'N', 'N', '#']3
POP: N
```

```
Entra: 4
estado q3:[('num', 'N', 'lam', 3), ('@', '#', 'lam', 4)]
Pila: ['N', '#']3
POP: N

Entra: @
estado q3:[('num', 'N', 'lam', 3), ('@', '#', 'lam', 4)]
POP: #
pila: []

Entra: a
estado q4:[('a', 'lam', 'lam', 5)]

Entra: l
estado q5:[('l', 'lam', 'lam', 6)]

Entra: u
estado q6:[('u', 'lam', 'lam', 7)]

Entra: m
estado q7:[('m', 'lam', 'lam', 8)]

Entra: n
estado q8:[('n', 'lam', 'lam', 9)]

Entra: o
estado q9:[('o', 'lam', 'lam', 10)]

Entra: s
estado q10:[('s', 'lam', 'lam', 11)]

Entra: .
estado q11:[('.', 'lam', 'lam', 12)]

Entra: u
estado q12:[('u', 'lam', 'lam', 13)]

Entra: d
estado q13:[('d', 'lam', 'lam', 14)]

Entra: g
estado q14:[('g', 'lam', 'lam', 15)]

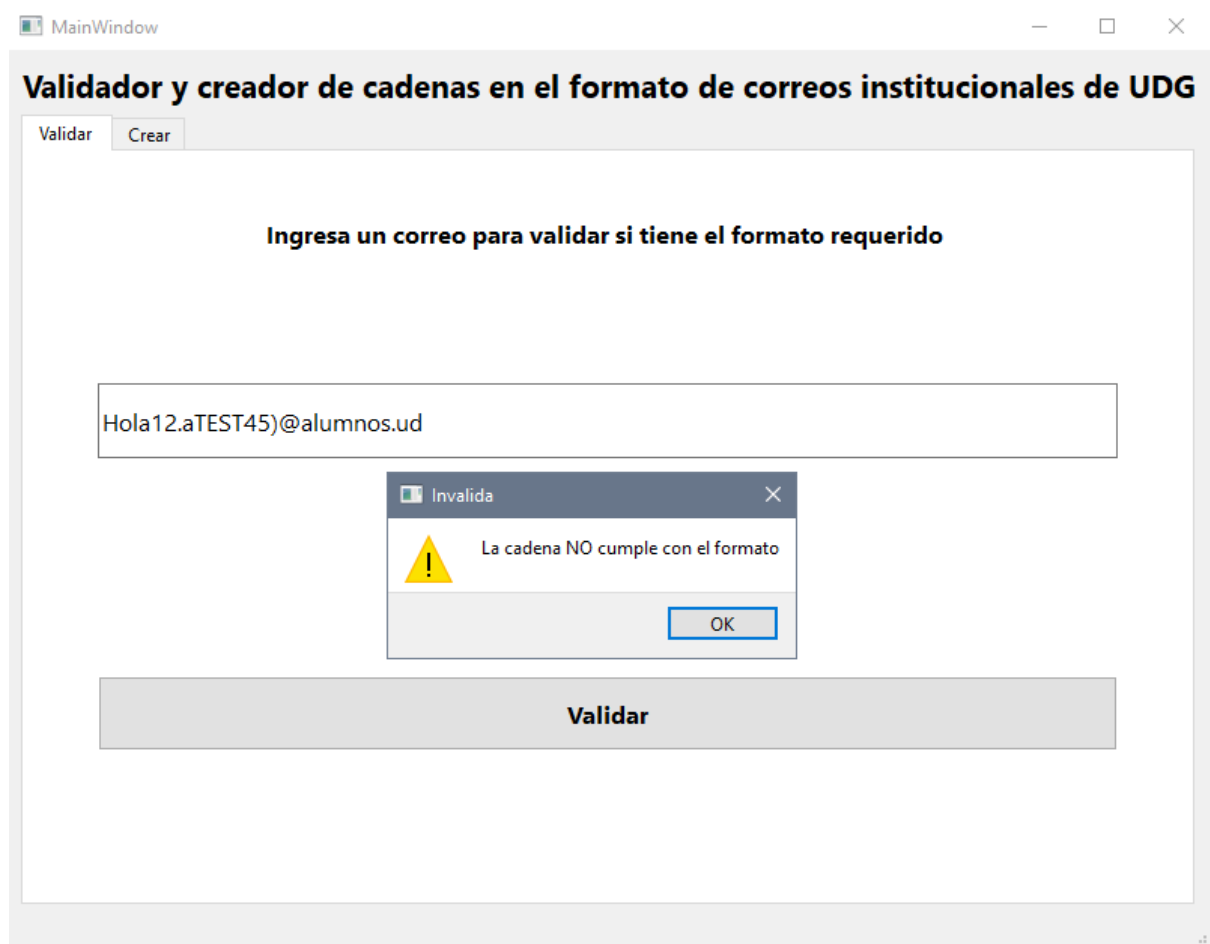
Entra: .
estado q15:[('.', 'lam', 'lam', 16)]

Entra: m
estado q16:[('m', 'lam', 'lam', 17)]

Entra: x
estado q17:[('x', 'lam', 'lam', 18)]
```

Es un proceso largo pero sirve para confirmar que el programa realiza de manera correcta cada paso igual que en el diagrama de estados; por otro lado si ingresamos

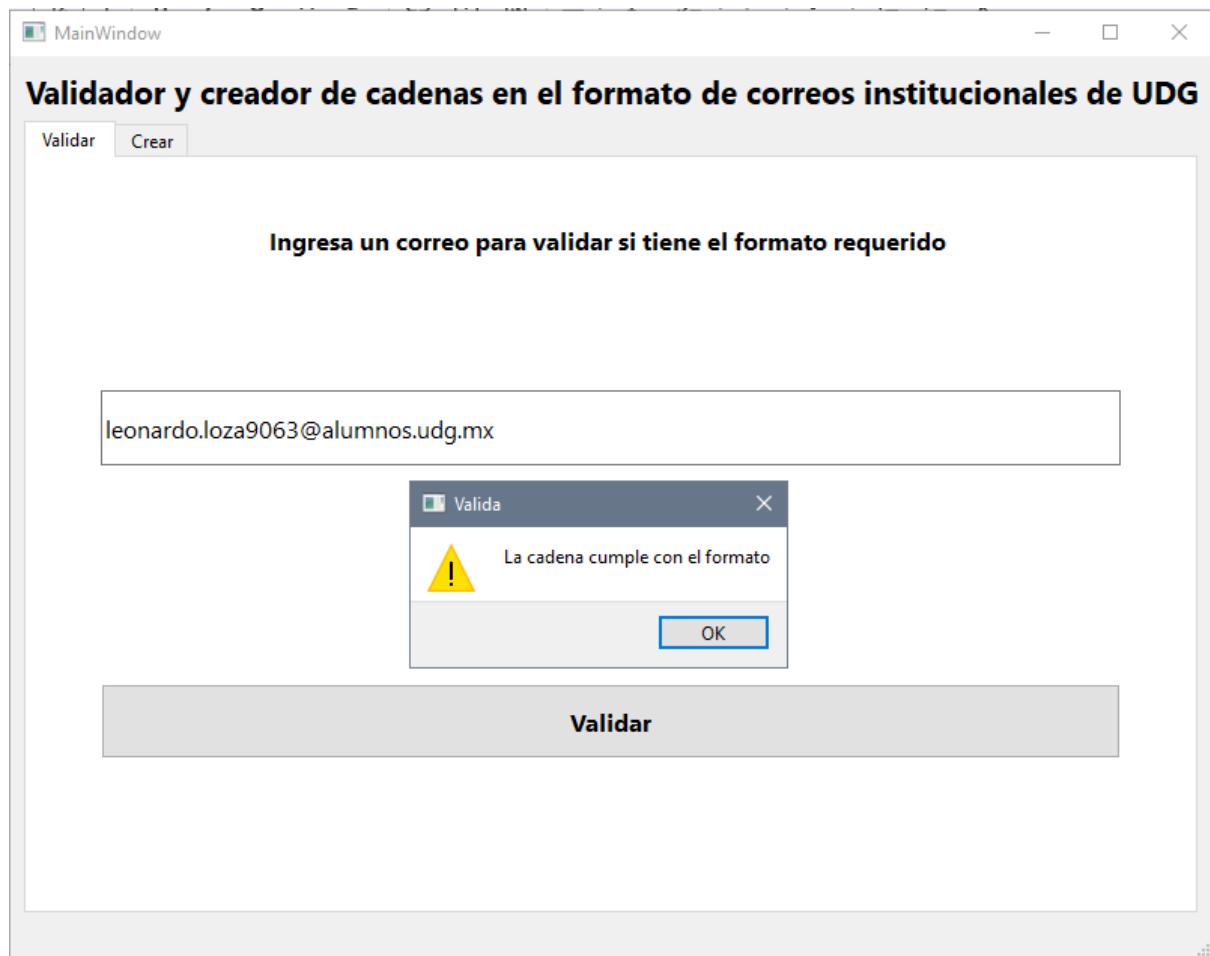
una cadena que no cumpla el formato, supere el número máximo de caracteres o comente un error en la parte final de la cadena saldrá una ventana emergente con el mensaje de que se invalida la cadena.



También en la consola podremos ver exactamente en qué estado se creó la no aceptación, en este caso desde el primer carácter se rechaza la cadena y se detiene todo el proceso.

```
Entra: H
estado q1:[('letMin', 'P', 'lam', 1), ('letMin', 'L', 'lam', 1), ('lam', 'L', 'lam', 1), ('.', 'lam', 'LLLLLLLLPPPP', 2)]
Pila: ['P', 'P', 'P', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', 'L', '#']1
Entra: o
```

Ahora probamos con otra cadena como el correo institucional del otro integrante del equipo "leonardo.loza9063@alumnos.udg.mx" donde observaremos que también se acepta la cadena.



Como extra notamos que con los primeros tres estrados de nuestro diagrama podríamos crear un algoritmo que generara correos institucionales recibiendo el nombre y apellido de cualquier alumno (todo en minúsculas).

MainWindow

Validador y creador de cadenas en el formato de correos institucionales de UDG

Validar Crear

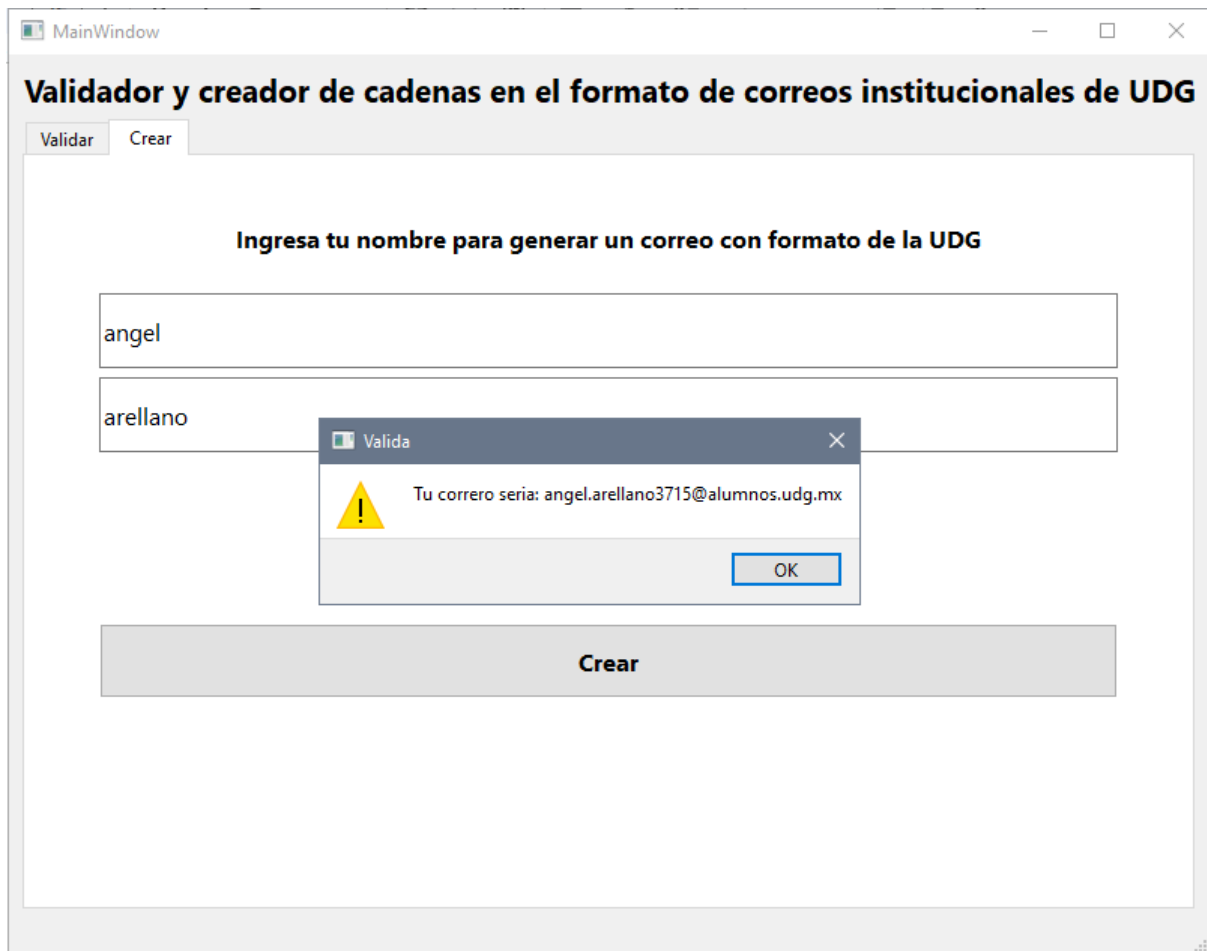
Ingresa tu nombre para generar un correo con formato de la UDG

Ingresa tu Nombre

Ingresa tu primer Apellido

Crear

Esto último es de lo que se encarga la otra pestaña del programa crear donde nos permite ingresar nuestro nombre y apellido para darle al botón que nos generaría un correo institucional, esto a través de una concatenación de ambas cadenas con un punto en el medio, donde si se acepta la sedena se concatenan cuatro números aleatorios y la constante '@alumnos.udg.mx' para mostrarla en una ventana emergente.



También puede no aceptarla por no cumplir con el formato requerido como usar otros caracteres no pertenecientes al abecedario en minúsculas.

MainWindow

Validador y creador de cadenas en el formato de correos institucionales de UDG

Validar Crear

Ingresa tu nombre para generar un correo con formato de la UDG

an@31

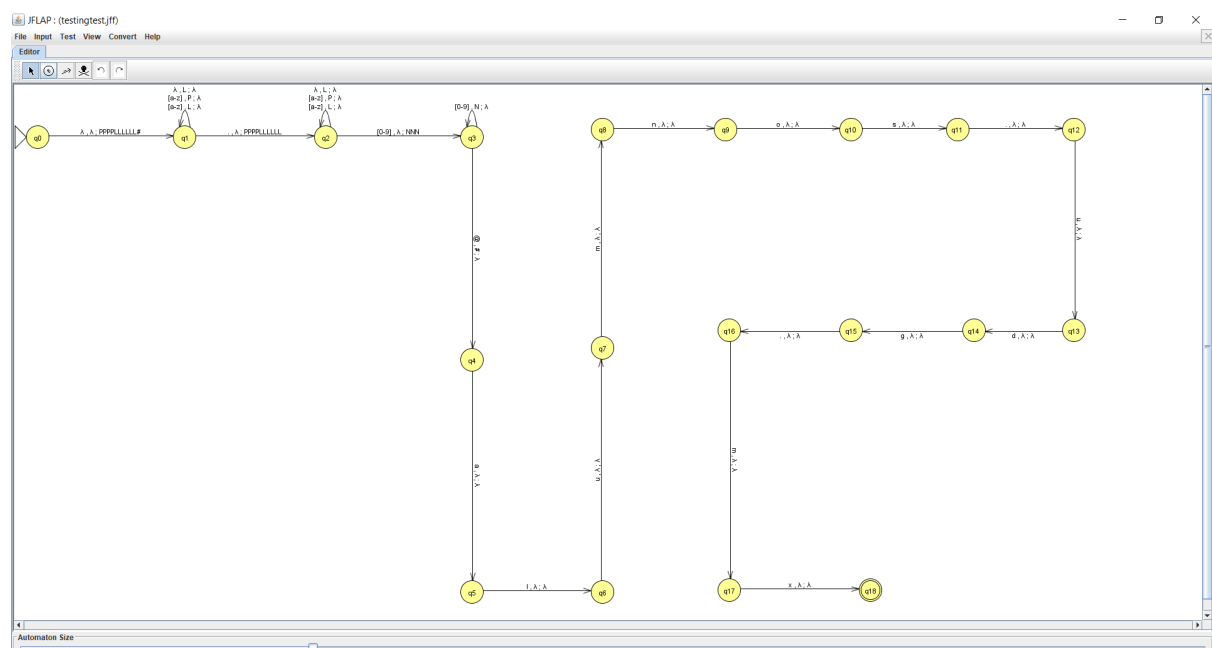
Kq23]

Invalida

Los datos NO cumplen con el formato

OK

Crear



Pruebas múltiples del autómata de pila

File Input Text View Convert Help

Editor Multiple Run

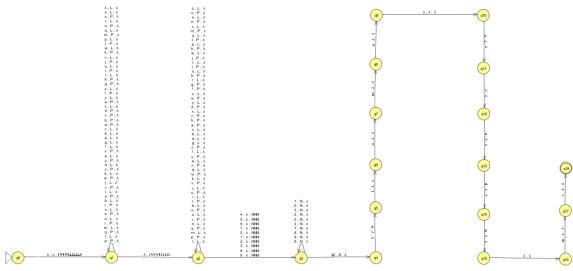


Table Text Size

Input	Result
eduardoesteban.montes1021@alumnos.udg.mx	Reject
enrique.rodriguez9012@alumnos.udg.mx	Accept
prueba.correo@hotmail.com	Reject
leonardo.loza4817@alumnos.udg.com	Reject
angel.altamirano@alumnos.udg.mx	Reject
sebastian7213@alumnos.udg.mx	Reject
andrea90.godinez1235@hotmail.com	Reject
correo.valido1253@alumnos.udg.mx	Accept
leo.lz1573@alumnos.udg.mx	Reject

Load Inputs Run Inputs Clear Enter Lambda View Trace

Conclusiones

En este proyecto de teoría de la computación, hemos utilizado el autómata de pila como herramienta para la validación de correos electrónicos institucionales.

La utilización del autómata de pila ha permitido mejorar la calidad y confiabilidad de la comunicación institucional al evitar errores comunes en las direcciones de correo electrónico, como caracteres inválidos o formatos incorrectos. Además, hemos podido implementar reglas adicionales para garantizar que las direcciones sean exclusivas de la institución, lo que contribuye a la seguridad y control en el uso del correo electrónico.

En conclusión, el uso del autómata de pila en la validación de correos electrónicos institucionales ofrece una solución confiable y eficiente para garantizar el formato correcto de las direcciones de correo electrónico. Este proyecto demuestra el poder de la teoría de la computación y los modelos formales para resolver problemas complejos en el ámbito de la comunicación institucional, y abre la puerta a futuras investigaciones y aplicaciones en este campo.

Referencias bibliográficas

Gómez Andrade, A. (2014). Introducción a la teoría de autómatas y lenguajes formales. TRAUCO.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2003). Introduction to Automata Theory, Languages, and Computation (2nd ed.). Addison-Wesley.

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed.). Prentice Hall.

Sipser, M. (2006). Introduction to the theory of computation (2nd ed.). Thomson Course Technology.