



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Prototipo de asistente corrector gramatical y ortográfico”

20-1-006

Presentan

Leonardo Miguel Aguirre Hernández

Zeth Álvarez Hernández

Director:

Dr. Miguel Santiago Suarez Castañón



Ciudad de México a 14 de noviembre de 2019



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO



No. de TT: 20-1-006

14 de noviembre de 2019

Documento Técnico

“Prototipo de asistente corrector gramatical y ortográfico”

Presentan:

Leonardo Miguel Aguirre Hernández¹

Zeth Álvarez Hernández²

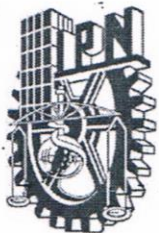
Director:

Dr. Miguel Santiago Suarez Castañón

Resumen – Este reporte presenta el avance de la propuesta e implementación de un prototipo de asistente corrector gramatical y ortográfico para la redacción de protocolos gramaticalmente correctos utilizando técnicas de procesamiento de lenguaje natural, todo esto basado en la arquitectura orientada a servicios, denominada software como servicio o SaaS por sus siglas en inglés.

Palabras clave –. Software como un servicio, corrección ortográfica y gramatical automatizada, machine learning, procesamiento de lenguaje natural.

¹ leonardomaguirreh@gmail.com, ² zethalvarezh@hotmail.com



ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA



**DEPARTAMENTO DE FORMACIÓN INTEGRAL E
INSTITUCIONAL**

COMISIÓN ACADÉMICA DE TRABAJO TERMINAL

México, D.F. a 14 de noviembre de 2019.

LIC. ANDRES ORTIGOZA CAMPOS
PRESIDENTE DE LA COMISIÓN ACADÉMICA
DE TRABAJO TERMINAL
P R E S E N T E

Por medio del presente, se informa que los alumnos que integran el **TRABAJO TERMINAL**: 20-1-006, titulado "Prototipo de asistente corrector gramatical y ortográfico" concluyeron satisfactoriamente su trabajo.

Los discos (DVDs) fueron revisados ampliamente por su servidor y corregidos, cubriendo el alcance y el objetivo planteados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

DR. MIGUEL SANTIAGO SUAREZ CASTAÑÓN
DIRECTOR DEL TRABAJO TERMINAL

Advertencia

“Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan.”

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52023.

Índice

Índice de figuras	7
Índice de tablas	8
Introducción.....	9
Capítulo 1. Problemática	11
1.1 Resumen del problema	12
1.2 Objetivos.....	12
1.2.1 Objetivo general	12
1.2.2 Objetivos particulares	12
1.3 Justificación	12
1.4 Estado del arte	13
Capítulo 2. Marco teórico y metodológico.....	15
2.1 Marco Teórico	16
2.1.1 La teoría lingüística de Noam Chomsky	16
2.1.2 RAE	16
2.1.3 Gramática.....	16
2.1.4 Ortografía	16
2.1.5 Oración	16
2.1.6 Estructura oracional.....	17
2.1.7 Inteligencia artificial.....	17
2.1.8 Machine learning	18
2.1.9 Procesamiento de lenguaje natural	18
2.1.10 Cómputo en la nube.....	18
2.1.11 Microsoft Azure.....	19
2.2 Marco Metodológico	20
2.2.1 Metodología Kanban	20
2.2.2 Patrón de diseño Health Endpoint Monitoring (monitoreo de puntos finales de salud).....	23
2.2.3 SOA, Service oriented architecture (Arquitectura orientada a servicios)	23
2.2.4 SaaS, Software as a Service (software como servicio).....	24
Capítulo 3. Análisis del sistema	25
3.1 Análisis del sistema	26

3.1.1 Reglas de negocio	26
3.1.2 Requisitos funcionales	26
3.1.3 Requisitos no funcionales	27
3.1.4 Casos de uso	27
3.1.5 Diagramas de secuencia.....	30
3.1.6 Entorno de desarrollo	33
3.1.7 Análisis de costos	33
3.1.8 Análisis de riesgos	35
Capítulo 4. Diseño	40
4.1 Resumen	41
4.2 Arquitectura	41
4.3 Desarrollo	41
4.4 Maquetado	48
4.5 Vista.....	49
4.6 Pruebas	50
Capítulo 5. Implementación	54
5.1 Resumen	55
5.2 Proceso	55
Conclusiones y trabajo a futuro	57
Conclusiones.....	58
Trabajo a futuro	58
Referencias	59
Glosario	61

Índice de figuras

Figura 1 Empresas de cómputo en la nube.....	18
Figura 2 Ejemplo de tablero Kanban.....	20
Figura 3 Tablero Kanban en Trello.....	22
Figura 4 Funcionamiento del patrón de diseño "Health Endpoint Monitoring".....	23
Figura 5 Diagrama de casos de uso.....	27
Figura 6 Diagrama de secuencia del caso de uso 1: detectar texto.....	30
Figura 7 Diagrama de secuencia del caso de uso 2: mostrar sugerencias.....	31
Figura 8 Diagrama de secuencia del caso de uso 3: corregir gramática.....	31
Figura 9 Diagrama de secuencia del caso de uso 4: corregir ortografía.....	32
Figura 10 Diagrama de secuencia del caso de uso 5: descargar archivos.....	32
Figura 11 Datos relevantes para estimación de costos.....	34
Figura 12 Tabla de estimaciones.....	34
Figura 13 Autómata para detección de errores entre mayúsculas y minúsculas.....	41
Figura 14 Ejemplo de pila.....	42
Figura 15 Búsqueda de palabras para sugerencias.....	44
Figura 16 Diccionario de palabras en español.....	45
Figura 17 Lista de palabras etiquetadas.....	46
Figura 18 Maquetado del prototipo.....	48
Figura 19 Vista del prototipo.....	49
Figura 20 Ejecución del algoritmo de búsqueda de palabras.....	50
Figura 21 Ejecución del algoritmo de corrección de mayúsculas, minúsculas y números.....	51
Figura 22 Ejecución del algoritmo de corrección de puntuación.....	51
Figura 23 Ejecución del algoritmo de corrección gramatical.....	52
Figura 24 Funcionamiento del sistema, parte 1.....	52
Figura 25 Funcionamiento del sistema, parte 2.....	53
Figura 26 Página de registro de Microsoft Azure.....	55
Figura 27 Inicio de Microsoft Azure.....	56

Índice de tablas

Tabla 1 Comparativa de aplicaciones.....	14
Tabla 2 Descripción de los servicios de Microsoft Azure.....	19
Tabla 3 Descripción del caso de uso 1.	28
Tabla 4 Descripción del caso de uso 2.	28
Tabla 5 Descripción del caso de uso 3.	29
Tabla 6 Descripción del caso de uso 4.	29
Tabla 7 Descripción del caso de uso 5.	30
Tabla 8 Tabla descriptiva de los niveles de probabilidad.....	36
Tabla 9 Tabla descriptiva de los niveles de impacto.	36
Tabla 10 Mapa de calor.	37
Tabla 11 Riesgos detectados.	38
Tabla 12 Descripción del plan de tratamiento de riesgos.	39
Tabla 13 Tabla de transición de estados.	44
Tabla 14 Tabla de estados de transición del autómata para gramática.....	47

Introducción

En la ESCOM, de acuerdo con una encuesta realizada en el mes de septiembre de 2019, aproximadamente el 45% de los alumnos piensan que las unidades de aprendizaje de formación institucional, como lo es Comunicación Oral y Escrita, son materias sin relevancia.

La falta de interés en este tipo de materias ocasiona que el alumno no adquiera adecuadamente los conocimientos necesarios para redactar de forma correcta un documento, lo cual, se ve reflejado en la pobre redacción de los diversos trabajos escritos (tareas, reportes, trámites, etc.) que realiza durante su estancia en la ESCOM.

Una mala redacción tiende a volverse una mala costumbre que afecta al alumno, principalmente al cursar unidades de aprendizaje como Trabajo Terminal, en la que la claridad y ortografía al escribir son de suma importancia.

Lo expuesto, nos motivó a desarrollar un prototipo de software que actúe como asistente para la redacción de protocolos gramatical y ortográficamente correctos. Para esto, utilizaremos técnicas de inteligencia artificial, en particular algoritmos de *machine learning* enfocados al procesamiento y análisis de textos, mejor conocidos como algoritmos de procesamiento de lenguaje natural (PLN), que nos permitirán detectar, en su mayoría, los errores ortográficos y gramaticales que se presenten. [2]

El software estará disponible como un servicio en la nube, con el objetivo de ponerlo a disposición del mayor número de usuarios posible. Disponer de un asistente en tiempo real, que ayude a pulir la escritura, además de ayudar a escribir correctamente, ayudará a conocer mejor las reglas gramaticales y ortográficas, porque el asistente al aconsejar, justificará la sugerencia señalando la o las reglas que se estarían infringiendo, o las que el asistente estaría aplicando.

Capítulo 1. Problemática

1.1 Resumen del problema

De acuerdo con una encuesta aplicada a 118 alumnos de la ESCOM, casi la mitad consideran que las unidades de aprendizaje de formación institucional no son relevantes y no les dan la importancia adecuada.

Al hacer esto, no adquieren los conocimientos necesarios para redactar correctamente, permitiendo que sus trámites y trabajos escolares se vean afectados, de tal modo que, al cursar unidades de aprendizaje en las que esto importa, entregan trabajos que carecen de calidad.

1.2 Objetivos

1.2.1 Objetivo general

- Realizar una herramienta que ayude a los estudiantes que estén cursando o hayan cursado el nivel superior de estudios a redactar correctamente protocolos en el idioma español (español de México).

1.2.2 Objetivos particulares

- Determinar qué métodos de procesamiento de lenguaje natural se deben implementar en el software.
- Realizar un algoritmo de corrección ortográfica.
- Realizar un algoritmo de corrección gramatical.
- Desplegar el asistente corrector en una plataforma orientada en SaaS utilizando una arquitectura SOA.

1.3 Justificación

Por lo presentado en la introducción, el poco interés de los alumnos en las unidades de aprendizaje de formación institucional, resulta un factor importante a considerar, pues se ve reflejado en su calidad para redactar correctamente.

Para una materia de gran importancia como lo es el trabajo terminal en la ESCOM, la calidad en la redacción de documentos es fundamental, ya que de esto depende que los profesores sinodales comprendan la idea principal del proyecto y al evaluar, no caigan en interpretaciones erróneas.

Es por eso que, el prototipo de este sistema está enfocado en desarrollar un asistente inteligente, que brinde ayuda en la corrección de errores ortográficos y gramaticales durante la redacción de los protocolos de Trabajo Terminal. Aunque, también se podrá usar en la redacción de cualquier otro texto. Con esto, además de mejorar la calidad del texto, se espera que proporcione sugerencias para que el usuario reciba una retroalimentación de sus errores.

Dada la gran cantidad de reglas ortográficas y gramaticales que contiene el idioma español, se tomaron en consideración 3 de las más usadas dentro de las siguientes áreas:

- Para la ortografía:
 - Uso de letras mayúsculas y minúsculas.
 - Signos de puntuación
 - 1 Punto
 - 2 Coma
 - 3 Guion
 - 4 Doble punto
 - 5 Punto y coma
 - 6 Paréntesis
 - 7 Doble comilla
 - 8 Signo de interrogación
 - 9 Signo de admiración
- Para la gramática:
 - Estructura básica de las oraciones.

Esto se decidió porque existe una gran pérdida en la calidad del lenguaje y una reducción en la capacidad de los estudiantes para redactar.

El prototipo, pretende ser una herramienta de apoyo que cubra la necesidad de corregir errores en la redacción de textos, en el idioma español de México. Ya que, por lo visto en el estado del arte, no existe una herramienta que cumpla adecuadamente con asistencia en este idioma.

Se implementarán algoritmos de PLN que lo harán independiente, esto quiere decir que, de alguna manera, el sistema reconocerá con base en ejemplos reales a distinguir y corregir errores ortográficos y gramaticales, sin la necesidad de un usuario que ingrese manualmente las reglas que rigen el idioma español.

1.4 Estado del arte

Una revisión exhaustiva de todas las aplicaciones existentes va más allá de los alcances de este trabajo terminal, por lo que hemos citado las que consideramos más importantes y que están relacionadas con la propuesta objeto de este documento:

- **Correctoronline.es:** herramienta para detectar errores ortográficos, y algunos otros básicos de gramática y estilo, en textos en español.
- **Ortógrafo:** corrector ortográfico que cuenta con el léxico más completo de este idioma (español) formado por más de cinco millones de palabras. Incluye términos técnicos de diferentes áreas como agricultura, biología y comercio, entre otras. Reconoce todas las formas conjugadas, así como diminutivos aumentativos, sufijos y prefijos más utilizados.
- **CorrectorOrtografico.com:** corrector ortográfico de más de 11 idiomas que permite la corrección de textos con hasta 1000 caracteres.
- **SpellBoy.com:** servicio que encuentra muchos errores que un simple corrector ortográfico no puede detectar. Además de los errores ortográficos normales, también detecta fallos gramaticales y estilísticos.
- **Corrector-Castellano.com:** corrector ortográfico y gramatical en español, ideal para escribir correctamente tus textos sin faltas de ortografía.

- **Spanishchecker.com:** servicio que detecta los errores cometidos por los estudiantes de español y ayuda a mejorar sus habilidades de escritura.

En la tabla 1, se muestra la comparativa de las aplicaciones que se describen arriba, contra el prototipo que desarrollaremos en nuestro trabajo terminal.

Servicio	Corrector ortográfico	Corrector gramatical	Corrector de puntuación	Corrección instantánea	Sugerencias de corrección	Español (México)	Sin costo
CorrectorOnline.es	✓	✗	✓	✗	✓	✗	✓
Ortógrafo (lenguaje.com)	✓	✗	✗	✗	✗	✗	✓
CorrectorOrtografico.com	✓	✗	✗	✗	✓	✗	✓
SpellBoy.com	✓	✓	✗	✗	✓	✗	✓
Corrector-castellano.com	✓	✓	✗	✗	✓	✗	✓
Spanishchecker.com	✓	✓	✗	✗	✓	✗	✗
Nosotros	✓	✓	✓	✓	✓	✓	✓

Tabla 1 Comparativa de aplicaciones.

Capítulo 2. Marco teórico y metodológico

2.1 Marco Teórico

A continuación se muestran los principales temas considerados para el desarrollo de este prototipo.

2.1.1 La teoría lingüística de Noam Chomsky

Se le conoce como la gramática generativa o biolingüística de Noam Chomsky. Postula la existencia de una estructura mental innata que permite la producción y comprensión de cualquier enunciado en cualquier idioma natural, posibilitando además que el proceso de adquisición de dominio del lenguaje hablado requiera muy poco conocimiento lingüístico para su correcto funcionamiento y se desarrolle de manera prácticamente automática. [3]

2.1.2 RAE

La Real Academia Española, «es una institución con personalidad jurídica propia que tiene como misión principal velar por que los cambios que experimente la lengua española en su constante adaptación a las necesidades de sus hablantes no quiebren la esencial unidad que mantiene en todo el ámbito hispánico», según establece el artículo primero de sus actuales estatutos. [4]

2.1.3 Gramática

El estudio de la gramática y la preparación de normas gramaticales han sido, desde los primeros estatutos académicos, un complemento imprescindible a la elaboración de diccionarios: en el diccionario se definen las palabras; en la gramática se explica la forma en que los elementos de la lengua se enlazan para formar textos y se analizan los significados de estas combinaciones. [5]

2.1.4 Ortografía

Según la RAE (Real Academia Española), la ortografía es el conjunto de normas que regulan la escritura de una lengua. [6]

2.1.5 Oración

La oración es la unidad máxima del análisis sintáctico. Se caracteriza por los dos rasgos siguientes:

- Es una unidad sintáctica formada por la unión de un predicado y su sujeto. Es decir, la oración constituye el marco sintáctico en el que se establece la relación predicativa.
- Posee necesariamente un verbo; salvo en las oraciones atributivas, este verbo constituye el núcleo del predicado. [7]

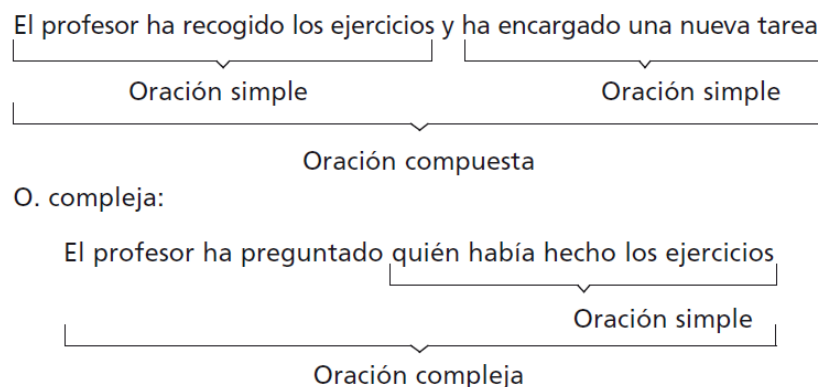
2.1.6 Estructura oracional

Las oraciones están formadas por elementos diversos relacionados entre sí. Las dos unidades sintácticas principales en las que se puede dividir una oración son el sujeto y el predicado, si bien la estructura interna de cada uno de estos constituyentes puede ser compleja. [7]

Clasificación de las oraciones

Las oraciones se clasifican atendiendo a diversos criterios. En primer lugar, la clasificación se basa en la estructura interna. Según ello, las oraciones se dividen en simples frente a complejas y compuestas. Una oración simple está formada por un único sujeto y un único predicado. Las oraciones compuestas o complejas, por el contrario, están formadas por la unión de más de una oración simple, de modo que poseen dos o más predicados con sus correspondientes sujetos. En una oración compuesta, las diversas oraciones que la forman mantienen entre sí una relación de igualdad jerárquica; en una oración compleja, en cambio, una de las oraciones está subordinada a la otra, de la que depende jerárquicamente. [7]

Ejemplo de oraciones simples, complejas y compuestas tomado del libro “Syntaxis, lengua española”.



2.1.7 Inteligencia artificial

Microsoft Azure, uno de los servicios más importantes de Microsoft en la actualidad, define la inteligencia artificial como "La capacidad de una máquina de imitar el comportamiento humano inteligente". [8]

Si bien el comportamiento humano es complejo, esta rama de la informática trata de crear programas y sistemas complejos que rivalicen y mejoren con las capacidades humanas para realizar tareas.

2.1.8 Machine learning

Andrew Ng., Profesor de la Universidad de Stanford define al Machine learning como “La ciencia de hacer que las computadoras actúen sin la necesidad de estar explícitamente programadas”. En cambio, expertos en Nvidia lo definen como “La práctica de usar algoritmos para analizar datos, aprender de estos y posteriormente hacer una determinación o predicción de algo”. [9]

Estas dos definiciones, así como otras varias que podemos encontrar en la red podemos decir que es dotar a las computadoras con programas que puedan analizar y aprender de los datos para realizar una tarea en específico.

2.1.9 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural o PLN es una disciplina ligada a lograr que las computadoras realicen tareas de utilidad que involucren el lenguaje humano y la lingüística. Es un campo multidisciplinario que se encarga de procesar las reglas de una lengua para mejorar las relaciones que involucren al humano y una máquina. [10]

De acuerdo a lo planteado, nuestro prototipo utilizara algoritmos de PLN para analizar textos, detectar errores en estos y dar correcciones gramatical y ortográficamente correctas.

2.1.10 Cómputo en la nube

De acuerdo con la Real Academia de Ingeniería, el cómputo en la nube es la utilización de las instalaciones propias de un servidor web albergadas por un proveedor de Internet para almacenar, desplegar y ejecutar aplicaciones a petición de los usuarios demandantes de las mismas. [11]

Actualmente, algunas de las empresas más destacadas por sus aportaciones al cómputo en la nube son: Microsoft, IBM, Google y Amazon.



Figura 1 Empresas de cómputo en la nube.

Tras una breve revisión de los servicios que brinda cada empresa, notamos que Microsoft Azure es el que más se acopla a este trabajo por las características de los servicios que brinda.

2.1.11 Microsoft Azure

Microsoft Azure es conjunto en constante expansión de servicios en la nube para ayudar a su organización a satisfacer sus necesidades comerciales. Otorga la libertad de crear, administrar e implementar aplicaciones en una red mundial enorme con sus herramientas y marcos favoritos. [12]

Microsoft Azure ofrece los servicios que se muestran en la tabla 2.

Implementación de la infraestructura <ul style="list-style-type: none"> • Máquinas virtuales Linux • Máquinas virtuales Windows • Azure Blueprints 	Protección y administración de recursos <ul style="list-style-type: none"> • Azure Security Center • Azure Monitor • Azure Applications Insights • Azure Cost Management • Azure Backup • Azure Site Recovery • Azure Migrate • Azure Policy
Desarrollo de aplicaciones <ul style="list-style-type: none"> • .NET • Python • Java • PHP • Node.js • Go 	Modelos de aplicaciones <ul style="list-style-type: none"> • Aplicaciones web • Funciones sin servidor • Contenedores • Microservicios con Kubernetes • Microservicios con Service Fabric
Administración de datos y IA (inteligencia artificial)	
Bases de datos relacionales <ul style="list-style-type: none"> • SQL Database como servicio • SQL Server en una máquina virtual de Azure • SQL Data Warehouse como servicio • Base de datos PostgreSQL como servicio 	IA y Servicios cognitivos <ul style="list-style-type: none"> • Machine Learning • Cognitive Services • Azure Notebooks
	NoSQL <ul style="list-style-type: none"> • Azure Cosmos DB
	Almacenamiento <ul style="list-style-type: none"> • Blob Storage

Tabla 2 Descripción de los servicios de Microsoft Azure.

2.2 Marco Metodológico

En esta sección se muestran la metodología, patrón de diseño y arquitectura que se utilizan en el desarrollo del prototipo.

2.2.1 Metodología Kanban

Kanban tiene prácticas que te permiten estabilizar y mejorar nuestro sistema para crear software.

Esta metodología se basa en el desarrollo incremental, dividiendo el trabajo en partes. Normalmente, cada una de esas partes se escribe en una nota y se pega en un tablero. Las notas suelen tener información variada, si bien, aparte de la descripción, debieran tener la estimación de la duración de la tarea. El tablero tiene tantas columnas como estados por los que puede pasar la tarea. [13]

Primero hay que conocer los principios fundamentales:

- Iniciar con lo que haces ahora.
- Estar de acuerdo en seguir un cambio incremental evolutivo.
- Al inicio, respetar los roles actuales, responsabilidades y títulos de trabajo.

Después adoptar las prácticas básicas:

- Visualizar.
- Limitar el trabajo en progreso.
- Gestionar el flujo.
- Realizar explícitas las políticas de proceso.
- Implementar ciclos de retroalimentación.
- Mejorar colaborativamente, evolucionar experimentalmente.

No se espera que las implementaciones adopten las seis prácticas inicialmente. Las implementaciones parciales se conocen como "poco profundas" con la expectativa de que aumenten gradualmente en profundidad a medida que se adoptan e implementan más prácticas con mayor capacidad. [14]

En la figura 2, se muestra un ejemplo del tablero de la metodología Kanban.

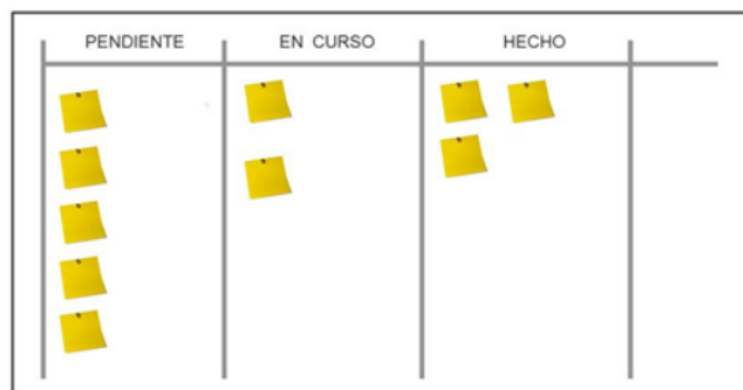


Figura 2 Ejemplo de tablero Kanban.

Algunas de las ventajas al implementar Kanban son:

- Facilidad de entendimiento.
- Visibilidad de la información a todos los miembros involucrados.
- Facilidad de integración con metodologías ágiles (Scrumban).
- El más adecuado para proyectos que se encuentran en mejora continua.
- Acepta el ingreso de cambios a último momento con facilidad.

En nuestro caso, en el desarrollo de este proyecto resulta bastante útil esta metodología, ya que permite que nos organicemos de tal forma que podemos enfocar nuestro tiempo principalmente en el área de desarrollo sin descuidar la documentación.

Por otra parte, dado que esta metodología tiene como requisito conocer los procesos de desarrollo y que, además, dentro de sus ventajas incluye la integración con otras metodologías ágiles, es necesario que ubiquemos cuáles son los procesos que manejaremos. Por lo que nos dimos a la tarea de implementar varios de los procesos que maneja la metodología ágil Scrum, tales como el listado de tareas priorizadas, la asignación de tareas para cada integrante, la asignación de fechas de entrega (fechas en que ya debe estar terminada la tarea), reuniones de revisión de tareas y reuniones planificación de tareas pendientes.

La figura 3, muestra cómo se usó la herramienta Trello para implementar esta metodología. En dicha figura se muestra la configuración de tablero Kanban utilizado que mejor se adaptó a nuestras necesidades y no permitía llevar un orden al momento de realizar tareas.

La distribución de las columnas y que representa cada una se muestra a continuación:

- Tareas (documentación): Nos proporciona información de las tareas pendientes relacionadas con la documentación del proyecto.
- Tareas (desarrollo): Nos proporciona información sobre las tareas pendientes relacionadas con el desarrollo del proyecto.
- En proceso (Max. 6 tareas) : Muestra las tareas que se están llevando a cabo en este momento. Solo tenemos permitido tener seis tareas en proceso de elaboración al mismo tiempo, esto es para evitar el acumulado de las mismas.
- En espera: Son tareas que se dejaron en pausa debido a que faltaban recursos o en ese momento es imposible terminarlas.
- Pendientes de revisión: Son tareas que si bien fueron finalizadas aun requieren que se revisen por parte de profesores sinodales, directores o los mismos miembros del equipo. Estas pueden regresar a “En proceso” si así se requiere.
- Hecho: Aquí movemos las tarjetas de las tareas que fueron finalizadas.

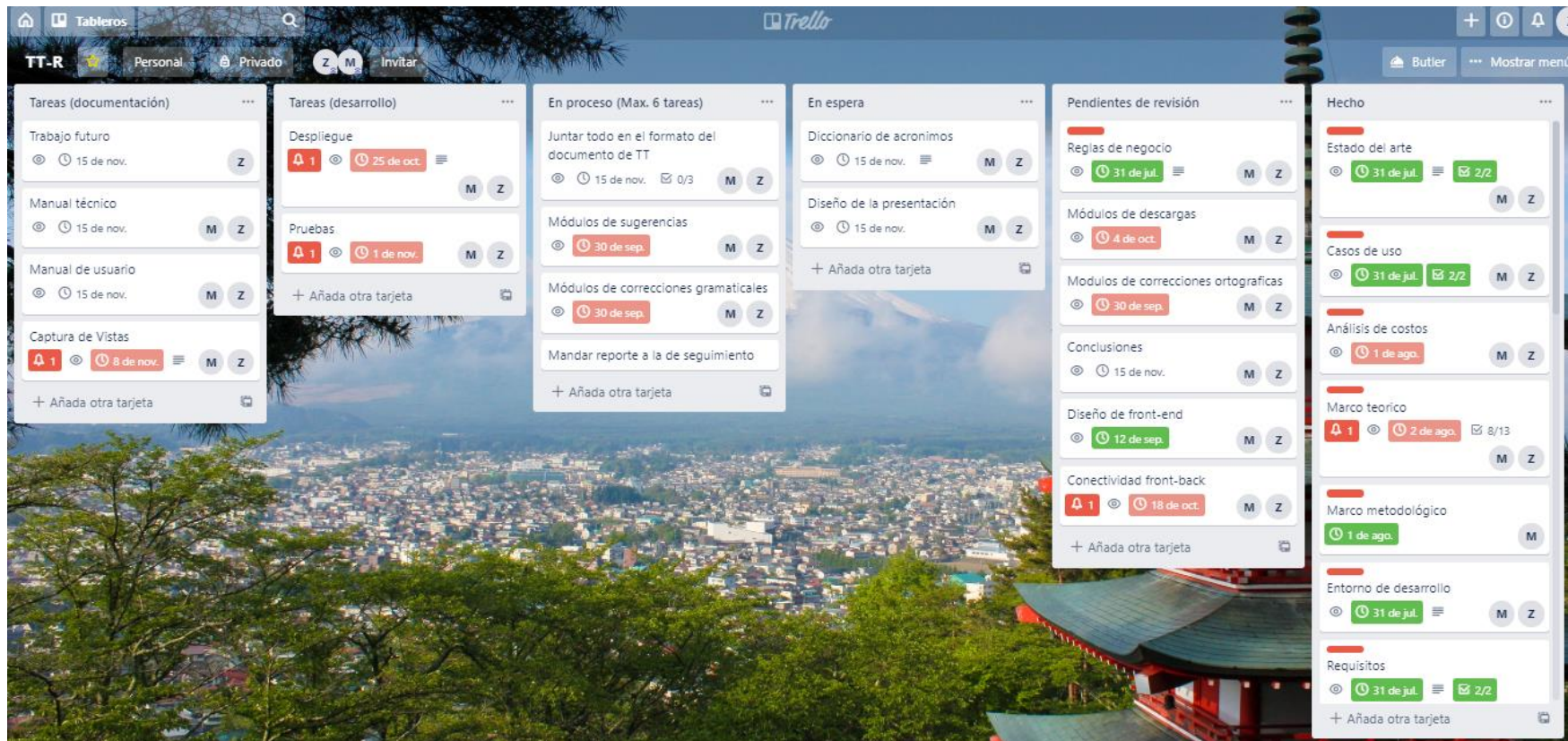


Figura 3 Tablero Kanban en Trello.

2.2.2 Patrón de diseño Health Endpoint Monitoring (monitoreo de puntos finales de salud)

Este proyecto es un prototipo de software que, por su funcionalidad, debe trabajar de tal forma que el usuario no deba cargar una y otra vez la misma ventana. Esto se vuelve una necesidad, que forzosamente requiere de una conexión asíncrona.

Por tal motivo, “Health Endpoint Monitoring” resulta el patrón de diseño adecuado para el desarrollo del prototipo, ya que implementa verificaciones funcionales en una aplicación a las que puedan acceder las herramientas externas a través de puntos finales expuestos a intervalos regulares. Esto puede ayudar a verificar que las aplicaciones y los servicios estén funcionando correctamente. [15]

Una verificación de monitoreo de salud típicamente combina dos factores:

- Las verificaciones (si las hay) realizadas por la aplicación o el servicio en respuesta a la solicitud al punto final de verificación de estado.
- Análisis de los resultados por la herramienta o marco que realiza la verificación de estado.

El código de respuesta indica el estado de la aplicación y, opcionalmente, cualquier componente o servicio que utiliza. La verificación de la latencia o el tiempo de respuesta se realiza mediante la herramienta de monitoreo o marco. De acuerdo con el sitio web de Microsoft, la figura 4 proporciona una visión general del patrón.

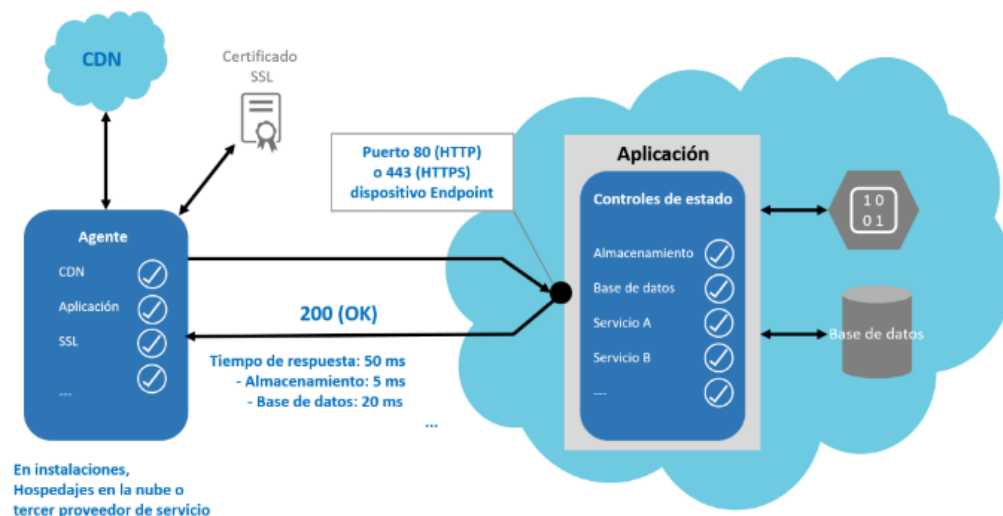


Figura 4 Funcionamiento del patrón de diseño "Health Endpoint Monitoring".

Otras comprobaciones que podrían ser realizadas por el código de monitoreo de salud en la aplicación incluyen:

- Verificación de almacenamiento en la nube o una base de datos para disponibilidad y tiempo de respuesta.
- Verificar otros recursos o servicios ubicados en la aplicación, o ubicados en otro lugar, pero utilizados por la aplicación.

2.2.3 SOA, Service oriented architecture (Arquitectura orientada a servicios)

SOA actualmente, significa que los componentes de una aplicación actúan como servicios interoperables y pueden ser usados de manera independiente y combinados con otros servicios. La implementación en contraste es considerada "software silo" (cualquier sistema de gestión que no pueda operar con ningún otro sistema), el cual raramente tiene interfaces de programación de aplicaciones (API) externalizables a componentes internos.

Si te equivocas respecto a lo que el cliente quiere en realidad, el costo es mucho menor con SOA que con "software silo" para recuperarse de los errores e intentar algo más o para producir una variante similar, pero no idéntica para complacer a un subconjunto de usuarios.

2.2.4 SaaS, Software as a Service (software como servicio)

El poder de SOA combinado con el poder de la Internet da espacio a un caso especial de SOA con su propio nombre: Software as a Service (SaaS). Ofrece software y datos como un servicio a través de la Internet, por lo general, a través de un programa como un navegador que se ejecuta en dispositivos cliente locales.

Las ventajas para el cliente y para el desarrollador de software son grandes:

1. Los clientes no necesitan instalar la aplicación, no tienen que preocuparse si su hardware es de la marca correcta o lo suficientemente rápido, ni si tienen la versión correcta del sistema operativo.
2. Los datos asociados con el servicio generalmente se guardan con el servicio, por lo que los clientes no deben preocuparse por hacer una copia de seguridad, perderlo debido a un mal funcionamiento del hardware local o incluso perder todo el dispositivo, como un teléfono o tableta.
3. Cuando un grupo de usuarios desea interactuar colectivamente con los mismos datos, SaaS es un vehículo natural.
4. Cuando los datos son grandes y / o se actualizan con frecuencia, puede tener más sentido centralizar los datos y ofrecer acceso remoto a través de SaaS.
5. Solo una copia del software del servidor se ejecuta en un entorno de sistema operativo y hardware uniforme y estrechamente controlado, seleccionado por el desarrollador, lo que evita los problemas de compatibilidad de la distribución de binarios que deben ejecutarse en computadoras y sistemas operativos de gran envergadura.
6. Los desarrolladores pueden probar nuevas versiones de la aplicación en una pequeña fracción de los clientes reales temporalmente sin molestar a la mayoría de los clientes.
7. Los desarrolladores no necesitan molestar a los usuarios con solicitudes de permiso para actualizar sus aplicaciones. [16]

Capítulo 3. Análisis del sistema

3.1 Análisis del sistema

Este capítulo muestra todas las reglas de negocio y requisitos que se consideraron para el desarrollo del sistema, además de los diagramas elaborados para su mejor comprensión.

3.1.1 Reglas de negocio

- RN1 Las correcciones están enfocadas al idioma español de México, conforme a la RAE y la Academia Mexicana de la Lengua.
- RN2 Los únicos caracteres que podrá detectar y analizar el prototipo serán aquellos que conforman la codificación UTF-8.
- RN3 No será necesario cargar la página para recibir las correcciones.
- RN4 Si la conexión se pierde, el sistema retoma la conexión sin necesidad de recargar la página.
- RN5 Las correcciones serán ortográficas y gramaticales.
- RN6 Los textos que se peguen no tendrán ningún formato.
- RN7 El servicio puede ser usado de forma ilimitada.
- RN8 El límite de caracteres será de 3500 caracteres.
- RN9 Si se llega al límite de caracteres (3500), el prototipo no dejará ingresar más.
- RN10 Los textos ingresados no se guardarán una vez que se vuelva a cargar página.
- RN11 La única forma de interactuar con el prototipo es mediante la ventana principal.

3.1.2 Requisitos funcionales

- RF1 El prototipo está enfocado a un solo tipo de usuario.
- RF2 Se podrán descargar los textos mediante archivos .pdf o .txt
- RF3 Se pueden copiar directo de la ventana los textos una vez corregidos.
- RF4 El prototipo debe realizar la detección de errores de forma asíncrona.
- RF5 Las sugerencias mostradas deben tener justificación.
- RF6 Las sugerencias modifican el texto únicamente cuando el usuario las elige.
- RF7 Realizar los dos tipos de correcciones y la generación de sugerencias dentro de una misma ventana.
- RF8 El ingreso de texto al prototipo será únicamente por medio de teclado y por medio de la función copiar-pegar.
- RF9 Estará disponible como un servicio en la nube.
- RF10 El prototipo realizará las correcciones después de que el usuario presione la tecla de punto o espacio.
- RF11 Indicar dónde y de qué tipo son los errores encontrados.
- RF12 Subrayar cada error detectado de diferente color de acuerdo al tipo de error.

3.1.3 Requisitos no funcionales

- RNF1 El prototipo se podrá ver en distintos dispositivo.
- RNF2 Para el desarrollo utilizaremos Python como lenguaje de programación, HTML, CSS y JavaScript como herramientas de desarrollo para front-end.
- RNF3 El prototipo estará disponible en la nube utilizando la herramienta Azure de Microsoft.
- RNF4 No se garantiza una corrección exacta si dentro del texto se incluyen modismos o abreviaturas en español o en algún otro idioma.

3.1.4 Casos de uso

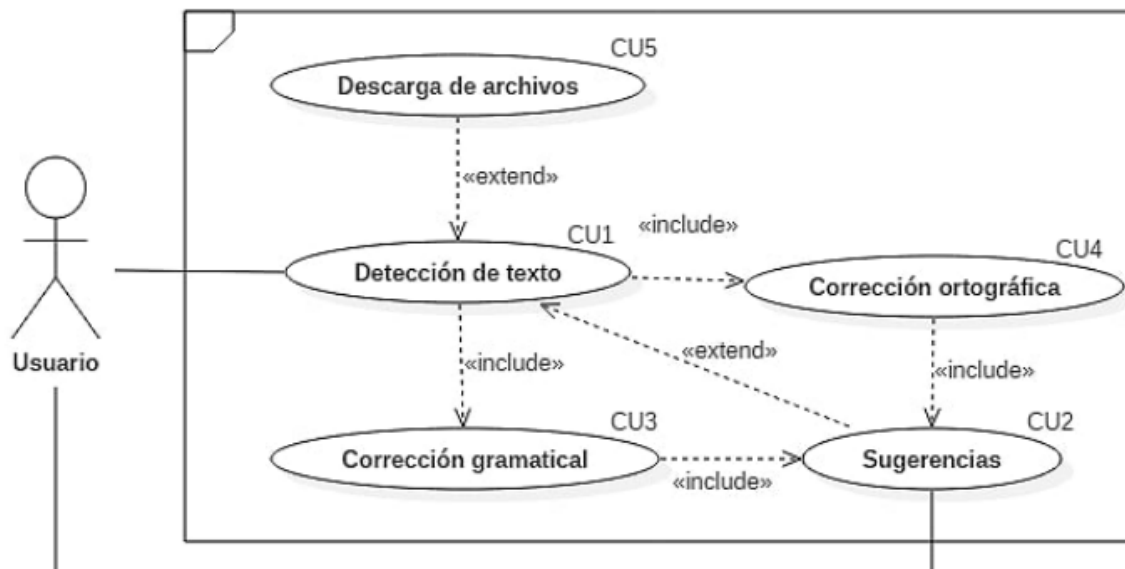


Figura 5 Diagrama de casos de uso.

Las siguientes tablas hacen referencia a los casos de uso mencionados en la figura 5:

Prototipo de asistente corrector ortográfico y gramatical	
Descripción del caso de uso 1	
Nombre	Detección de texto
Identificador	CU1
Actores	Usuario
Función	Recibir el texto que ingresa el usuario mediante el teclado o por medio de la función copiar-pegar.
Entradas	Texto.
Salidas	Visualización del texto ingresado.
Precondición	Ingresar al servicio.
Postcondición	Se puede trabajar un texto nuevo en la misma ventana.
Referencias	RF3, RF8, RN1, RN2, RN6, RN8, RN9, RN11.
Escenario principal	Se carga la página del servicio y se ingresa el texto que se desea corregir.

Tabla 3 Descripción del caso de uso 1.

Prototipo de asistente corrector ortográfico y gramatical	
Descripción del caso de uso 2	
Nombre	Sugerencias
Identificador	CU2
Actores	Usuario
Función	Apoyándose del CU3 y CU4 se cargarán las posibles sugerencias junto con su respectiva justificación y se mostrarán al usuario como opciones de reemplazo.
Entradas	Errores encontrados con CU3 y CU4.
Salidas	Sugerencias y justificaciones según el error identificado.
Precondición	Haber detectado errores en el texto por medio de CU3 y/o CU4.
Postcondición	Mostrar las sugerencias y sus respectivas justificaciones para los errores previamente detectados.
Referencias	RN1, RN3, RN4, RF4, RF5, RF6, RF7, RF11.
Escenario principal	Al colocar el cursor sobre el error, CU2 se encargará de mostrar las posibles sugerencias para corregirlo.

Tabla 4 Descripción del caso de uso 2.

Prototipo de asistente corrector ortográfico y gramatical	
Descripción del caso de uso 3	
Nombre	Corrección gramatical
Identificador	CU3
Actores	Usuario
Función	A partir del CU1, los algoritmos de PLN realizarán la detección de errores gramaticales del texto ingresado por el usuario.
Entradas	Texto ingresado por el usuario.
Salidas	Errores gramaticales detectados en el texto.
Precondición	Haber detectado texto en CU1.
Postcondición	Enviar los errores encontrados a CU2 y marcarlos en el texto.
Referencias	RN1, RN5, RF4, RF7, RF10, RF11, RF12.
Escenario principal	Una vez ingresado el texto, se procesará por medio de los algoritmos de PLN y se determinará si hay errores gramaticales.

Tabla 5 Descripción del caso de uso 3.

Prototipo de asistente corrector ortográfico y gramatical	
Descripción del caso de uso 4	
Nombre	Corrección ortográfica
Identificador	CU4
Actores	Usuario
Función	A partir del CU1, los algoritmos de PLN realizarán la detección de errores ortográficos del texto ingresado por el usuario.
Entradas	Texto ingresado por el usuario.
Salidas	Errores ortográficos detectados en el texto.
Precondición	Haber detectado texto en CU1.
Postcondición	Enviar los errores encontrados a CU2 y marcarlos en el texto.
Referencias	RN1, RN5, RF4, RF7, RF10, RF11, RF12.
Escenario principal	Una vez ingresado el texto, se procesará por medio de los algoritmos de PLN y se determinará si hay errores ortográficos.

Tabla 6 Descripción del caso de uso 4.

Caso de uso 5

Prototipo de asistente corrector ortográfico y gramatical	
Descripción del caso de uso 5	
Nombre	Descarga de archivos
Identificador	CU5
Actores	Usuario
Función	Permitirá descargar el texto trabajado mediante un archivo en formato .pdf o .txt.
Entradas	Opción de formato para descargar el contenido.
Salidas	Archivo en formato .pdf o .txt.
Precondición	Haber detectado texto en CU1.
Postcondición	Se puede descargar el contenido que se encuentra en el área de texto las veces que el usuario desee siempre y cuando no actualice la página.
Referencias	RN11, RF2.
Escenario principal	Una vez terminadas las correcciones, el usuario tendrá la opción para descargar el texto corregido.

Tabla 7 Descripción del caso de uso 5.

3.1.5 Diagramas de secuencia

A continuación, de la figura 6 a la figura 10 se muestran los diagramas de secuencia de los casos de uso 1 a 5 respectivamente.

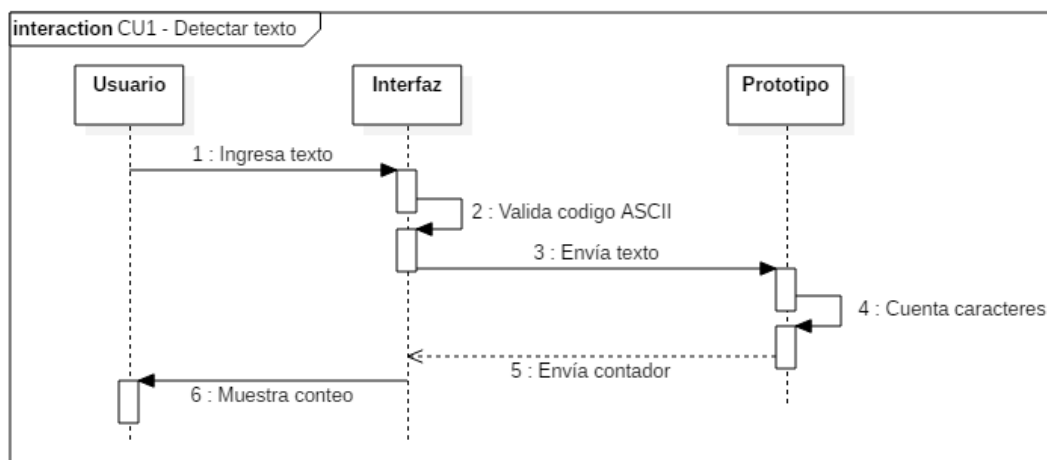


Figura 6 Diagrama de secuencia del caso de uso 1: detectar texto.

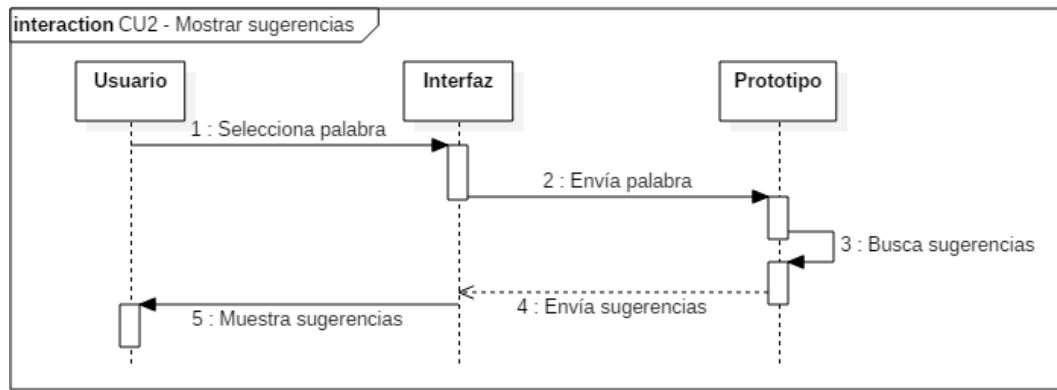


Figura 7 Diagrama de secuencia del caso de uso 2: mostrar sugerencias.

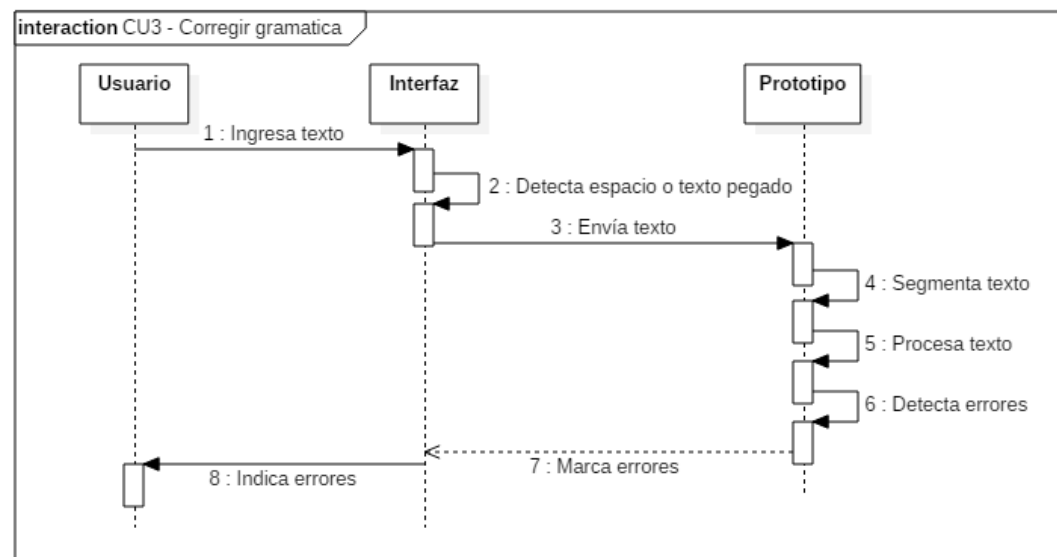


Figura 8 Diagrama de secuencia del caso de uso 3: corregir gramática.

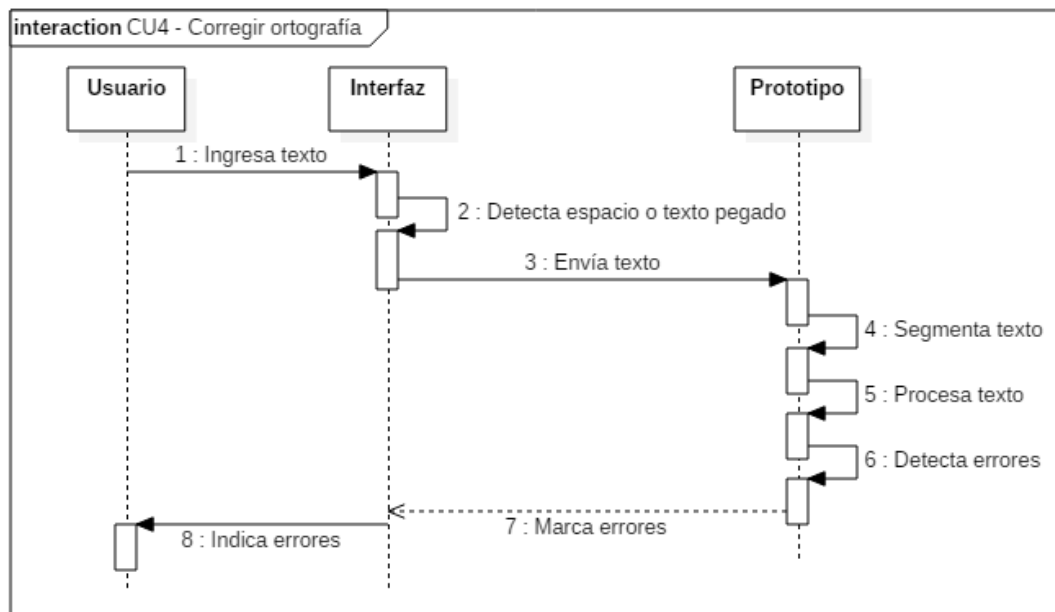


Figura 9 Diagrama de secuencia del caso de uso 4: corregir ortografía.

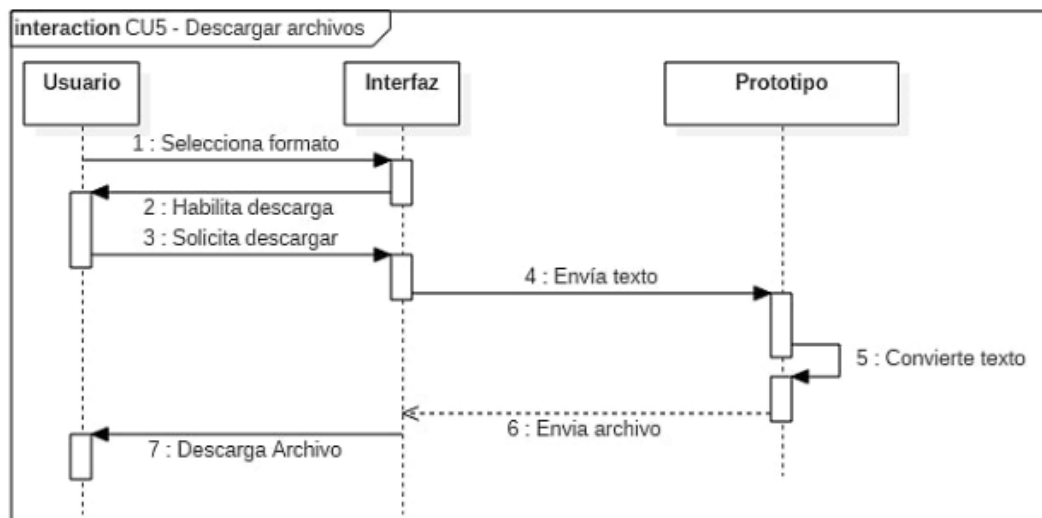


Figura 10 Diagrama de secuencia del caso de uso 5: descargar archivos.

3.1.6 Entorno de desarrollo

De acuerdo con la investigación previamente realizada y por cuestiones de compatibilidad, decidimos usar las siguientes tecnologías para desarrollar este prototipo:

Lenguaje de programación

- Python
- JavaScript

Herramientas de desarrollo de front-end

- HTML5
- CSS3

Servicio de cómputo en la nube

- Microsoft Azure

Maquetado

- Ninja Mock

Organizador para metodología

- Trello

Almacenamiento en la nube

- OneDrive
- Google Drive

IDEs

- Jupyter notebook
- PyCharm
- Visual Studio

3.1.7 Análisis de costos

Generalidades

Para realizar el análisis de costos de este prototipo nos pusimos a analizar diversas técnicas para conseguir un análisis consistente y lo más apegado a la realidad. Esto nos llevó a los análisis COCOMO en especial el COCOMO II.

El Modelo Constructivo de Costes COCOMO (Constructive Cost Model) es utilizado en proyectos de software para estimar los costes del mismo en función de tres submodelos: básico, intermedio y detallado.

En modelo COCOMO es uno de los sistemas de estimación de costes más utilizados en proyectos de desarrollo de software. Si bien la estandarización de este modelo por su fácil uso y aplicación nos da aproximaciones muy reales al coste final del proyecto, optamos por utilizar una variante de dicho modelo, el COCOMO II.

Esto se planteó así debido a que ya existen herramientas que nos realizan los cálculos sin la necesidad de meternos con fórmulas y valores constantes dados para dicho modelo, únicamente nos es requerido el total de líneas de código y una investigación sobre el sueldo promedio de un ingeniero en sistemas, que para este caso fue de \$11,771 MXN al mes. [17]

Se utilizó la herramienta USC-COCOMO II.2000.4 la cual nos permite hacer la estimación por Conversión de Puntos Función a Líneas de Código Fuente o SLOC. En la figura 14 se aprecian los datos obtenidos de la imagen anterior ya dentro de la herramienta.

Si bien debemos realizar algunas conversiones para que la herramienta trabaje bien, esto no genera alguna complejidad. La herramienta USC-COCOMO II utiliza dólares para algunos de sus cálculos por lo que adaptamos el campo de *LABOR Rate* de acuerdo al tipo de cambio actual del sueldo promedio investigado con anterioridad. Este valor pasa a ser de \$613.12 USD.

Como observación, el sistema maneja dólares por lo que hicimos la conversión correspondiente y con el tipo de cambio actual rellenamos el campo *LABOR Rate* la cual corresponde a \$615.09 USD dólares americanos.

Project Name: Scale Factor: Schedule

Development Model:

X	Module Name	Module Size	LABOR Rate (\$/month)	EAF	Language	NOM Effort DEV	EST Effort DEV	PROD	COST	INST COST	Staff	RISK
	TT-R-20-1-006	S:3000	613.12	1.00	Non-Specified	9.8	9.8	304.8	6033.69	2.0	1.3	0.0

Figura 11 Datos relevantes para estimación de costos.

La tabla de la figura 15 nos muestra los resultados calculados para todos los módulos combinados. Incluye el tamaño total del proyecto en líneas de código (Total Lines of Code) y una tabla con una estimación *optimista* (Optimistic), otra *más probable* (Most Likely) y otra *pesimista* (Pessimistic) para el esfuerzo (Effort), la duración (Sched), la productividad (PROD), el coste (COST), el coste por instrucción (INST), el personal necesario (Staff) y el riesgo (RISK, de éste sólo se muestra el valor más probable).

Ya con los datos ingresados al sistema, este nos arrojará valores catalogados entre los campos *optimista*, *más probable* y *pesimista*, como se muestra en la figura 15. A continuación se describen los resultados obtenidos.

Total Lines of Code:	3000	Estimated	Effort	Sched	PROD	COST	INST	Staff	RISK
Hours/PM:	152.00	Optimistic	7.9	7.1	381.1	4826.95	1.6	1.1	
		Most Likely	9.8	7.6	304.8	6033.69	2.0	1.3	0.0
		Pessimistic	12.3	8.2	243.9	7542.11	2.5	1.5	

Figura 12 Tabla de estimaciones.

Tabla de estimaciones

Optimistas:

En el caso optimista vemos que los campos más relevantes nos muestran que se deberá realizar en una duración de 7.1 días para conseguir realizar el proyecto, con una persona basta para terminarlo y el costo del proyecto será de \$4,826.95 dólares americanos.

Mas probable:

En el caso más probable se requerirá un esfuerzo de 9.8 para la realización, las personas involucradas de igual manera será una y el costo de realización asendereará a \$6,033.69 dólares americanos.

Pesimista:

Siendo pesimistas el esfuerzo, así como el tiempo invertido aumentará a 8.2 días y un monto respectivamente. Serán requeridas personas y media (dos personas) para finalizar el sistema y el costo aumentará a \$7,542.11 dólares americanos.

3.1.8 Análisis de riesgos

Generalidades

El análisis del riesgo implica desarrollar una comprensión del riesgo. Este análisis consiste en determinar las consecuencias y sus probabilidades para sucesos de riesgo identificados antes del desarrollo de algún proyecto.

El análisis de riesgo considera las causas y fuentes del riesgo, sus consecuencias, y la probabilidad de que estas consecuencias puedan ocurrir. Se deberían identificar los factores que afectan a las consecuencias y a la probabilidad.

Este análisis incluye normalmente una estimación de la gama de posibles consecuencias que se podrían derivar de un suceso, situación o circunstancia asociando sus probabilidades de ocurrencia con el nivel de impacto que tienen en el desarrollo del proyecto. Para llevar a cabo esta asociación utilizaremos la fórmula presentada a continuación la cual es un producto del nivel de probabilidad de ocurrencia y el nivel de impacto.

$$\text{Nivel de riesgo} = \text{Nivel de probabilidad} \times \text{Nivel de impacto}$$

Fórmula 1 Fórmula para el cálculo del nivel de riesgo.

Los métodos que se utilizan en el análisis de riesgos pueden ser cualitativos, semicuantitativos o cuantitativos. El grado de detalle requerido dependerá de la aplicación particular, de la disponibilidad de datos fiables y de la necesidad de tomar decisiones durante el desarrollo.

En la tabla 8 se describen las escalas cuantitativas y cualitativas del nivel de probabilidad de ocurrencia de sucesos que representen riesgos para el desarrollo de Prototipo de asistente corrector gramatical y ortográfico.

Nivel de probabilidad

Nivel	Descripción	Concepto
5	Casi cierto	La amenaza tiene una muy alta probabilidad de ocurrir durante el desarrollo del proyecto. Es casi seguro que suceda durante el desarrollo.
4	Muy frecuente	Es probable que suceda esta amenaza durante el desarrollo del proyecto.
3	Frecuente	Se tiene una probabilidad media de ocurrir con frecuencia. De unas 3 a 8 veces durante el desarrollo.
2	Ocasional	Es poco probable que suceda esta amenaza durante el desarrollo del proyecto.
1	Rara vez	La amenaza tiene una probabilidad muy baja de ocurrir durante el desarrollo del proyecto.

Tabla 8 Tabla descriptiva de los niveles de probabilidad.

A su vez, en la tabla 9 describimos los parámetros utilizados para medir el nivel de impacto que tienen los sucesos durante el desarrollo de este proyecto.

Nivel de impacto

Nivel	Descripción	Concepto
5	Muy grave	Si el evento llegara a presentarse, tendría un trágico impacto, comprometiendo los objetivos de la empresa o del proyecto.
4	Grave	Si el evento llegara a presentarse, tendrá un alto impacto, comprometiendo los objetivos de la empresa, del proyecto o paralización del flujo de trabajo principal.
3	Moderado	Si el evento llegara a presentarse, tendrá un impacto moderado, afectando los objetivos del proyecto y el flujo de trabajo en procesos de soporte.
2	Bajo	Si el evento llegara a presentarse, tendrá un bajo impacto, sobre algunas de las actividades o tareas dentro del flujo de trabajo.
1	Muy bajo	Este evento si se llegara a presentar, no representa un impacto importante para el desarrollo.

Tabla 9 Tabla descriptiva de los niveles de impacto.

Mapa de calor

Con base a las escalas de niveles de probabilidad e impacto generamos un mapa de calor utilizando la fórmula 1, vista al inicio del capítulo. A su vez, definimos cuales son las acciones que se deben de tomar dependiendo el nivel de riesgo.

		PROBABILIDAD				
		Rara vez 1	Ocasional 2	Frecuente 3	Muy frecuente 4	Casi cierto 5
IMPACTO	Muy grave 5	Medio 5	Alto 10	Alto 15	Muy alto 20	Muy alto 25
	Grave 4	Medio 4	Medio 8	Alto 12	Alto 16	Muy alto 20
	Moderado 3	Bajo/Muy bajo 3	Medio 6	Alto 9	Alto 12	Alto 15
	Bajo 2	Bajo/Muy bajo 2	Medio 4	Medio 6	Medio 8	Medio 10
	Muy bajo 1	Bajo/Muy bajo 1	Bajo/Muy bajo 2	Bajo/Muy bajo 3	Medio 4	Medio 5

Tabla 10 Mapa de calor.

- Nivel de riesgo: Muy alto – Factor de riesgo:

$$FR \geq 20$$

Riesgo por encima del nivel de riesgo aceptable de la organización. **Requiere acciones inmediatas y valorar la posibilidad de parar la actividad que origina el riesgo.** Se incluye en el plan de tratamiento de riesgos. Medidas preventivas obligatorias.

- Nivel de riesgo: Alto – Factor de riesgo:

$$9 \leq FR < 20$$

Riesgo elevado para el proyecto. Se debe realizar seguimiento y evaluación periódica de la eficacia de las acciones. Se incluye en el plan de tratamiento de riesgos. Medias preventivas obligatorias.

- Nivel de riesgo: Medio – Factor de riesgo:

$$4 \leq FR < 9$$

Riesgo medio. Antes de comenzar con el desarrollo del proyecto se deberá decidir si se deben incluir en el plan de tratamiento de riesgos, acciones o medidas preventivas para reducir el nivel de riesgo. Si no fuera posible, mantener las variables controladas.

- Nivel de riesgo: Bajo/Muy bajo – Factor de riesgo:

$$FR < 4$$

Riesgo por debajo del nivel de riesgos aceptables. Se vigilará, aunque no requiere acciones o medidas preventivas obligatorias.

Identificación de los riesgos

Tras una lluvia de ideas, llegamos a la conclusión de que los riesgos expuestos en la tabla 11 son los más propensos a ocurrir, y en los cuales, un análisis de riesgos nos servirá para tener un plan de contingencia para prever riesgos y afectar lo menos posible el flujo de trabajo.

Identificador	Evento amenazante (Riesgo)	Factor	Nivel de probabilidad	Nivel de impacto	Nivel de riesgo
R1	Abandono del proyecto por parte de algún miembro del equipo de trabajo.	Interno	1	5	5
R2	Detectar riesgos durante el desarrollo.	Interno	2	4	8
R3	Cambios de último minuto por parte de sinodales o directores. (Clientes)	Externo	2	4	8
R4	Implementación incorrecta del algoritmo de sugerencias.	Interno	2	5	10
R5	Implementación incorrecta del algoritmo de corrección ortográfica.	Interno	2	5	10
R6	Implementación incorrecta del algoritmo de corrección gramatical.	Interno	2	5	10
R7	Acumulación de tareas en el tablero Kanban.	Interno	4	2	8
R8	Falla del servicio.	Externo	1	4	4
R9	Curva de aprendizaje en el uso de las herramientas para programar ML y PLN.	Interno	3	4	12
R10	Cambio de fechas de presentación.	Externo	1	2	2
R11	Cambio de fechas y horas para revisión de avances.	Externo	3	2	6
R12	Retrasos en el desarrollo de algún modulo del prototipo.	Interno	3	4	12

Tabla 11 Riesgos detectados.

De acuerdo con los valores que obtuvimos en la columna *Nivel de riesgo*, estos representan según su color el factor de riesgo y las acciones que debemos tomar en cuenta para anticiparnos a esta situación. Para este caso únicamente realizaremos un plan de tratamiento para las situaciones con un factor de riesgo *alto* y *muy alto*, ya que las situaciones catalogadas con un factor de *medio* no representan un problema que necesite acciones inmediatas.

Plan de tratamiento de riesgos

Identificador	Riesgo	Propuesta de tratamiento (Plan de acción)
R4	Implementación incorrecta del algoritmo de sugerencias.	Utilizar tiempos muertos y agendar un espacio para repasar el porqué de los errores ocurridos. Si es necesario recalendarizar tareas para volver a programar el módulo y realizar pruebas nuevamente.
R5	Implementación incorrecta del algoritmo de corrección ortográfica.	Utilizar tiempos muertos y agendar un espacio para repasar el porqué de los errores ocurridos. Si es necesario recalendarizar tareas para volver a programar el módulo y realizar pruebas nuevamente
R6	Implementación incorrecta del algoritmo de corrección gramatical.	Utilizar tiempos muertos y agendar un espacio para repasar el porqué de los errores ocurridos. Si es necesario recalendarizar tareas para volver a programar el módulo y realizar pruebas nuevamente
R9	Curva de aprendizaje en el uso de las herramientas para programar ML y PLN.	Antes de comenzar con la programación de los módulos realizar una reunión donde cada miembro del equipo de trabajo expresara sus dudas respecto a las herramientas a utilizar. De ser necesario se deberán recalendarizar algunas tareas para repasar o aprender dichas herramientas.
R12	Retrasos en el desarrollo de algún modulo del prototipo.	Posponer tareas que no afecten el flujo principal de trabajo. De ser necesario dividir la tarea entre los miembros del equipo que hayan terminado sus asignaciones. Las tareas que componen cada módulo, así como los módulos en si deberán ser programados con un margen de tiempo para revisar, hacer pruebas y tener un “colchón de tiempo” en caso de retrasos.

Tabla 12 Descripción del plan de tratamiento de riesgos.

Capítulo 4. Diseño

4.1 Resumen

Este capítulo muestra detalladamente cómo fue desarrollado el prototipo.

4.2 Arquitectura

Como se menciona en el marco teórico, la arquitectura SOA permite que este prototipo sea orientado como un servicio, además de que al ser un prototipo, tendrá constantes modificaciones, cosa que con SOA resulta bastante sencillo. Por otra parte, éste será un software que ofrece sus servicios a través de la Internet, lo cual abre paso al uso del caso particular de SOA llamado SaaS.

4.3 Desarrollo

Para que este prototipo funcione, es necesario que se detecten errores ortográficos y gramaticales, para el caso de los errores ortográficos, se utilizó lo siguiente:

- **Autómatas:** para la detección de errores en el uso de mayúsculas, minúsculas y números, ya que indicarán cuándo está mal implementado el uso de alguna letra o número. Por otra parte, también resultan útiles en la detección de errores de puntuación.
- **Pilas:** para identificar cuando un signo de interrogación, exclamación, paréntesis o comilla doble no se ha cerrado o abierto (según sea el caso).

A continuación, en la figura 16, se muestra el autómata implementado para la detección de errores en mayúsculas y minúsculas.

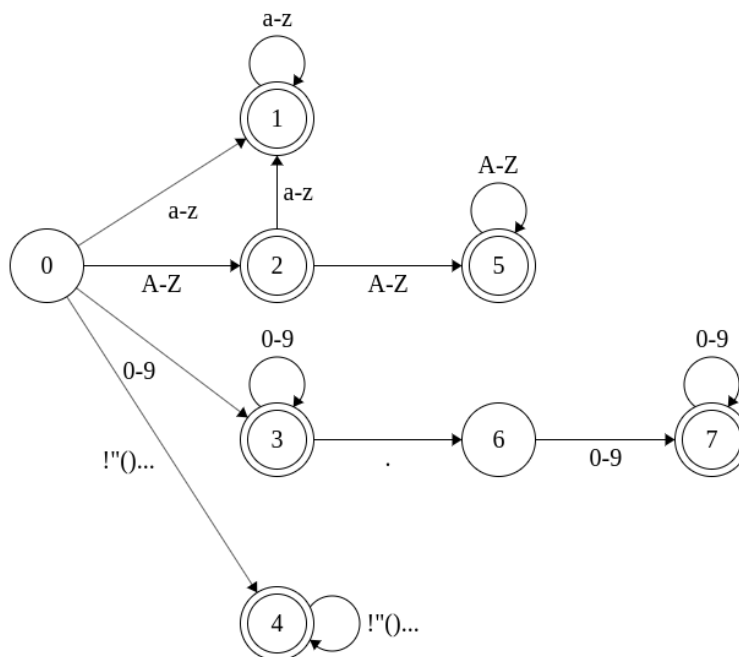


Figura 13 Autómata para detección de errores entre mayúsculas y minúsculas.

Para el caso de los signos de puntuación, nos dimos a la tarea de separar en dos partes la detección de errores:

1. Detección de apertura y cierre de signos, para el caso del paréntesis, la doble comilla y los signos de interrogación y admiración.
2. Orden de implementación, para los antes mencionados y la coma, el punto, el guion, los dos puntos y el punto y coma.

En el primer caso, se hizo uso de una pila, de la cual, el funcionamiento es el siguiente:

1. Si el signo es $\rightarrow (; \rightarrow$ apilar.
2. Si el signo es $\rightarrow) ? ! \rightarrow$ buscar su opuesto en la pila.
3. Si la pila no contiene el opuesto, enviar el signo a estado de error.
4. Si el último signo de la pila es el opuesto, desapilar y descartar ambos signos.
5. Si el último signo de la pila no es el opuesto, desapilar, mandarlo a estado de error y regresar al paso 2.
6. Si el signo es $\rightarrow " \rightarrow$ preguntar si la pila contiene uno igual.
7. Si la pila contiene $\rightarrow " \rightarrow$ ir al paso 4.
8. Si se recorrió todo el texto y la pila no está vacía, mandar el contenido a estado de error.

La figura 17, muestra un ejemplo de pila como la implementada.

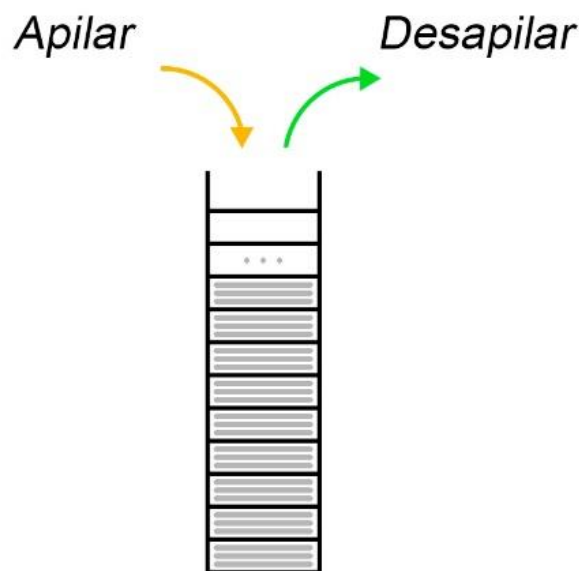


Figura 14 Ejemplo de pila.

Para el segundo caso, dado que el autómata resulta complicado de dibujar, en la tabla 13 se muestra la transición de estados.

S26	E	E	E	E	E	E	E	E	E	E	E	{S27, S28, S29}	E	E	E	E
S27	{S27}	E	E	{S6}	E	E	E	E	E	E	E	E	E	E	E	{S5, S6, s11}
S28	{S27}	{S28}	E	{S6}	E	E	E	E	E	E	E	E	E	E	E	{S5, S6, s11}
S29	E	E	{S29}	{S6}	E	E	E	E	E	E	E	E	E	E	E	{S5, S6, s11}

Tabla 13 Tabla de transición de estados.

Búsqueda de palabras para sugerencias: Para realizar la búsqueda de las sugerencias de las palabras se optó por una búsqueda tipo *divide y vencerás* la cual se explica su funcionamiento a continuación:

Primero se accede al archivo correspondiente a la palabra de la cual se quiere buscar la sugerencia, esto se logra extrayendo el tamaño de la cadena y la letra inicial de esta. Con estos datos es posible asignar en un arreglo todas las palabras de dicho archivo y comenzar la búsqueda.

La búsqueda comienza asignando un *pivote* el cual en primera instancia se ubica en la posición central del archivo de palabras. Posteriormente se pregunta si el carácter siguiente de la palabra a buscar tiene un valor mayor o menor a la letra de la palabra en la que está el *pivote*, si es menor se asigna un nuevo pivote teniendo como referencia para la nueva mitad el *pivote* anterior como máximo, para caso contrario, si el carácter comparado es mayor el pivote se moverá hacia una mitad de abajo tomando como mínimo el valor del *pivote* actual. Esto se ejemplifica en la figura X.

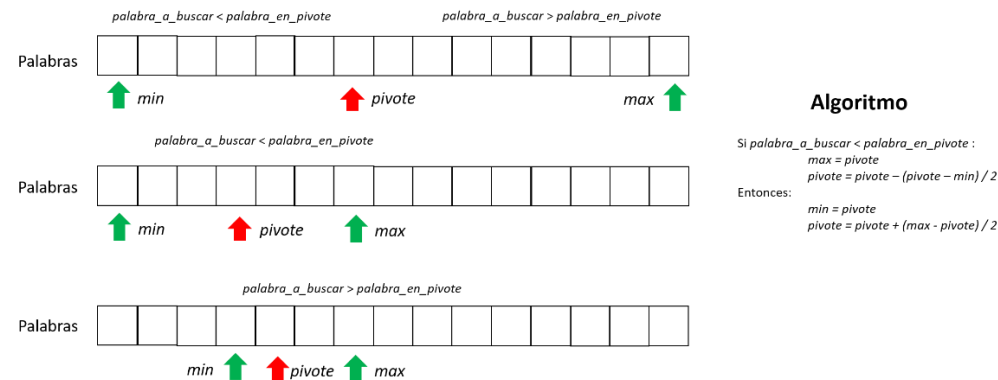


Figura 15 Búsqueda de palabras para sugerencias.

Segmentación de diccionarios: Dado que el diccionario de palabras ocupado para la verificación de las palabras constaba de aproximadamente 67800 palabras, se buscó una forma óptima de buscar palabras sin forzar a utilizar una gran cantidad de memoria y procesamiento como si lo hiciéramos por fuerza bruta. Para este algoritmo se optó por segmentar el diccionario en pequeños archivos, cada uno dividido por letra inicial de la palabra y numero de caracteres como se muestra en la figura 18 un ejemplo de dichos archivos.

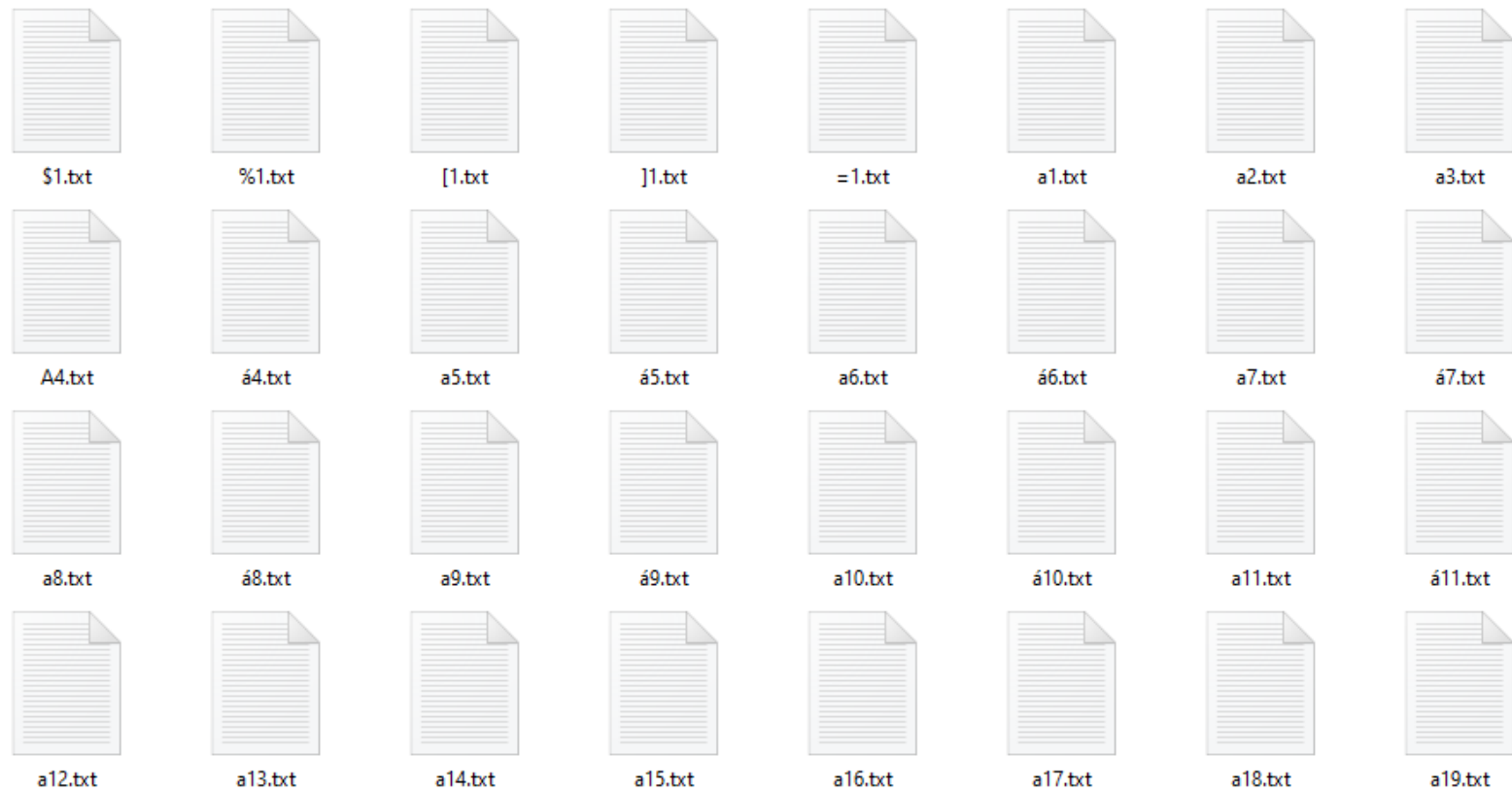


Figura 16 Diccionario de palabras en español.

Para la parte gramatical, nos enfocamos en la estructura de las oraciones. Como se muestra en el marco teórico, las oraciones se dividen en tres categorías: simples, compuestas y complejas; pero toda oración contempla elementos básicos fundamentales, el sujeto, verbo y predicado. A simple vista se puede manejar como sujeto y predicado, pero para funcionalidad de este proyecto, se catalogaron en 7 tipos de palabras y una palabra estática:

- A – Complementos
- B – Sujetos
- C – Verbos
- D – Adjetivos
- E – Adverbios
- F – Conectores
- G – No (palabra)
- H – Preguntas (qué, cómo, cuándo, dónde, quién, cuánto, etc.)

Para poder identificar las palabras mediante esta clasificación, se etiquetaron manualmente alrededor de 3500 palabras. La figura 17 muestra algunas de las palabras que ya fueron etiquetadas.

SaaS/B
a/F
arquitectura/B
asistente/B
avance/B
basado/F
como/F
corrector/B
correctos/D
de/F
de/F
de/F
de/F
denominada/C
e/F
el/A
en/F
en/F
este/A
gramatical/D
gramaticalmente/D
implementación/B

Figura 17 Lista de palabras etiquetadas.

Estas etiquetas se usaron en como reglas de producción para un autómata, el cual determina si la estructura de la oración es correcta o incorrecta.

El autómata cuenta con 13 estados y 8 transiciones que se muestran a continuación.

	A	B	C	D	E	F	G	H
→ 0	1	2	7	12	6	1	11	10
1	-	2	-	-	-	-	-	-
2*	1	2	3	5	-	4	9	-
3*	1	2	3	5	-	-	9	-
4*	1	2	3	5	-	4	-	-
5*	1	2	-	5	-	4	9	-
6	1	2	7	-	-	-	8	2
7*	1	2	7	4	-	-	-	10
8	-	-	7	-	-	-	-	-
9*	-	-	3	-	-	-	-	-
10	1	2	7	-	-	-	8	-
11	-	-	7	-	-	-	-	-
12	1	2	-	12	-	-	-	-

Tabla 14 Tabla de estados de transición del autómata para gramática.

4.4 Maquetado

Para este prototipo, contamos únicamente con una sola ventana dado que solamente se considera un único tipo de usuario. La figura 19 muestra el maquetado del prototipo.

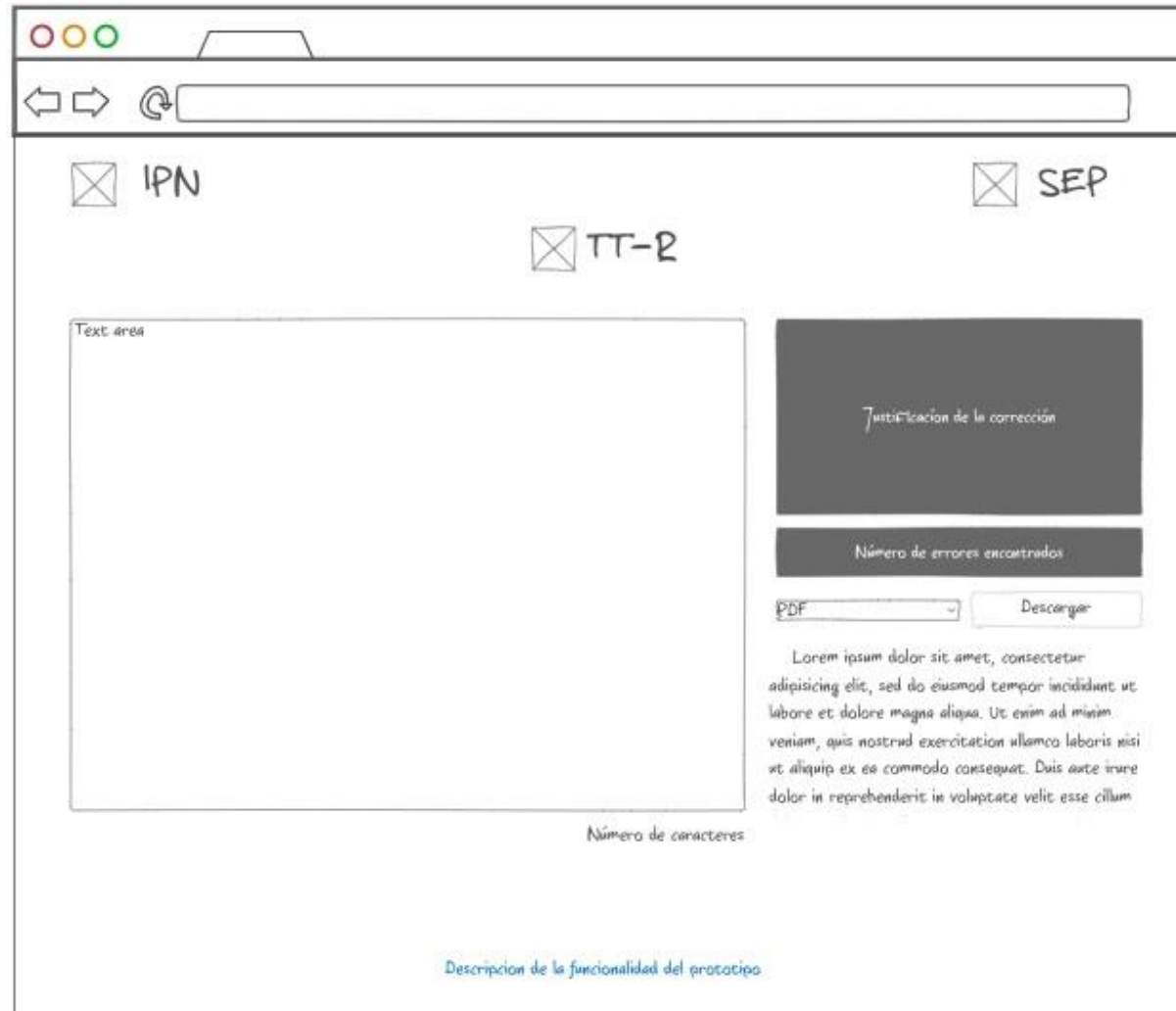


Figura 18 Maquetado del prototipo.

4.5 Vista

Para fines prácticos, la vista de este prototipo se desarrolló de tal forma que el usuario no tenga complicaciones para utilizarlo, la figura 20 es una captura de pantalla de la vista actual del sistema.

The screenshot displays a web application interface for a grammar and spelling correction assistant. At the top, there is a header with three logos: the Mexican coat of arms, the text 'GOBIERNO DE MÉXICO', and the text 'EDUCACIÓN SECRETARÍA DE EDUCACIÓN PÚBLICA'. To the right of these is the logo of the 'Instituto Politécnico Nacional' with the tagline 'La Técnica al Servicio de la Patria'. Below the header, the title 'Prototipo de asistente corrector gramatical y ortográfico' is centered. On the right side, the text 'Trabajo terminal remedial 20-1-006' is visible. The main area features a large text input box with the placeholder 'Escribe o pega (Ctrl+V) tu texto aquí...'. Above this box, on the right, is a label 'Número de caracteres: 0'. To the right of the input box is a section titled 'Sugerencias'. Below the input box, there is a 'Descargar' button. Underneath the button, a message states 'Puedes descargar tu texto corregido aquí.' followed by a dropdown menu currently showing 'Archivo de texto (.txt)'. At the bottom right, another 'Descargar' button is present.

Figura 19 Vista del prototipo.

4.6 Pruebas

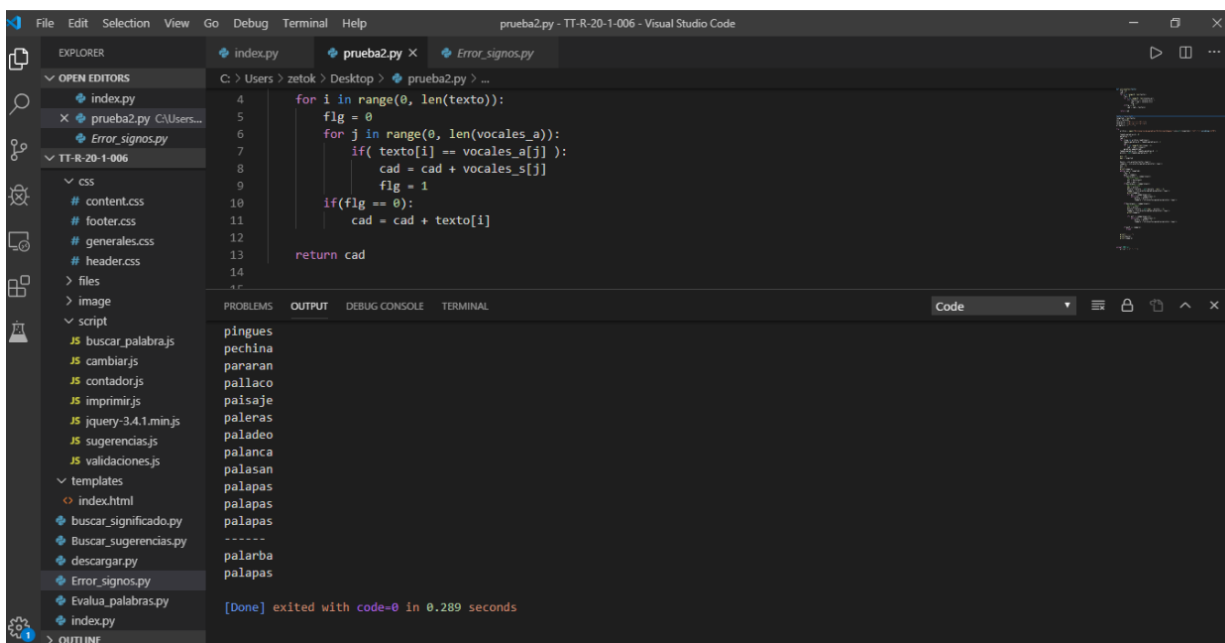
Con el fin de proporcionar información clara sobre la calidad del prototipo de asistente corrector gramatical y ortográfico, realizamos pruebas de software sobre este.

Las pruebas son un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento durante el desarrollo y deberán ser tratadas como el código del sistema en sí, con documentación y con el mismo nivel de profesionalismo.

Pruebas unitarias

Estas consisten en probar cada función, clase y modulo del sistema individualmente. Cada prueba debe ser independiente del resto y deberán cumplir con ciertas características para asegurar la calidad de estas.

Cada uno de los algoritmos funcionan correctamente por separado, en las figuras 20, 21, 22 y 23 se muestra la ejecución de cada algoritmo por separado.



```
File Edit Selection View Go Debug Terminal Help
prueba2.py - TT-R-20-1-006 - Visual Studio Code

EXPLORER
OPEN EDITORS
index.py
prueba2.py C:\Users\...
Error_signos.py
TT-R-20-1-006
css
content.css
footer.css
generales.css
header.css
files
image
script
buscar_palabras.js
cambiar.js
contador.js
imprimir.js
jquery-3.4.1.min.js
sugerencias.js
validaciones.js
templates
index.html
buscar_significado.py
Buscar_sugerencias.py
descargar.py
Error_signos.py
Evalua_palabras.py
index.py
OUTLINE

prueba2.py
4 for i in range(0, len(texto)):
5     flg = 0
6     for j in range(0, len(vocales_a)):
7         if( texto[i] == vocales_a[j] ):
8             cad = cad + vocales_s[j]
9             flg = 1
10        if(flg == 0):
11            cad = cad + texto[i]
12
13    return cad
14

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code

píngues
pechina
pararan
pallaco
paisaje
paleras
paladeo
palanca
palasan
palapas
palapas
palapas
-----
palarba
palapas

[Done] exited with code=0 in 0.289 seconds
```

Figura 20 Ejecución del algoritmo de búsqueda de palabras.

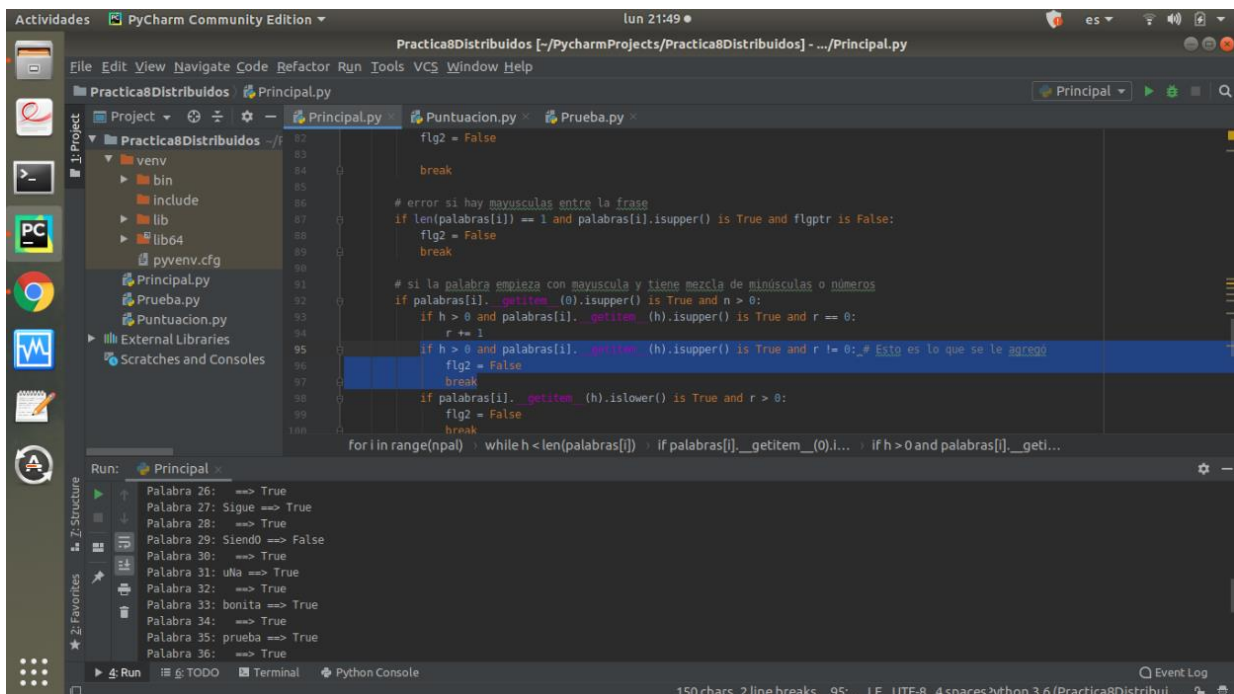


Figura 21 Ejecución del algoritmo de corrección de mayúsculas, minúsculas y números.

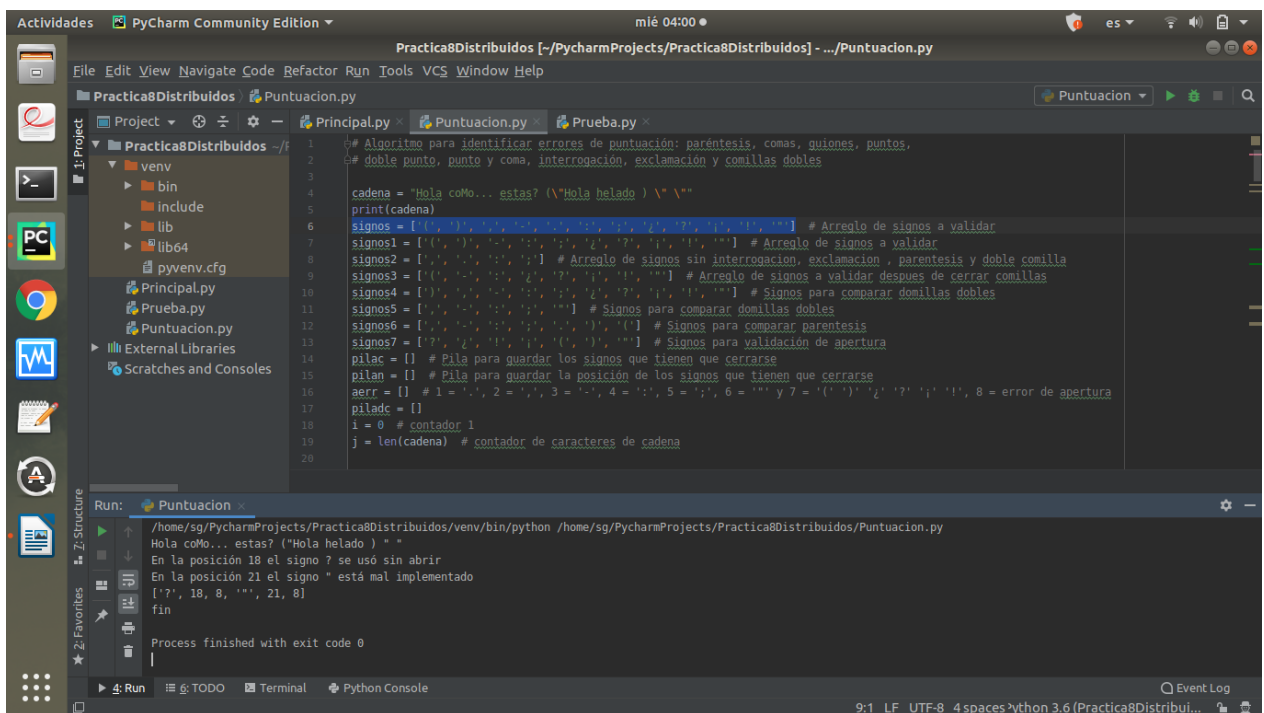


Figura 22 Ejecución del algoritmo de corrección de puntuación.

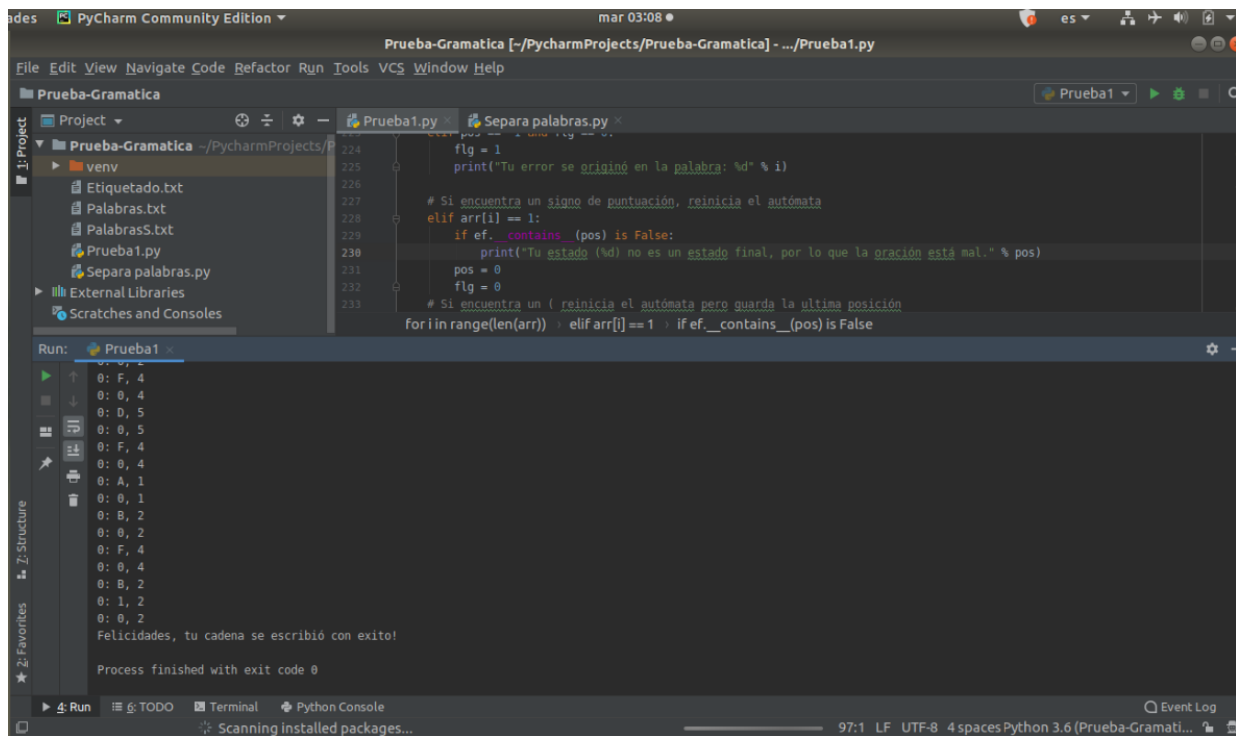


Figura 23 Ejecución del algoritmo de corrección gramatical.

Pruebas de integración

Una vez completadas las pruebas unitarias se deberán probar los elementos individuales del sistema como un todo. Los módulos deberán funcionar correctamente trabajando en conjunto.

Posteriormente, una vez que se probaron los algoritmos por separado, se juntaron los módulos para realizar pruebas de su funcionamiento, el cual resultó exitoso, como se ve en la figura 24 y 25.

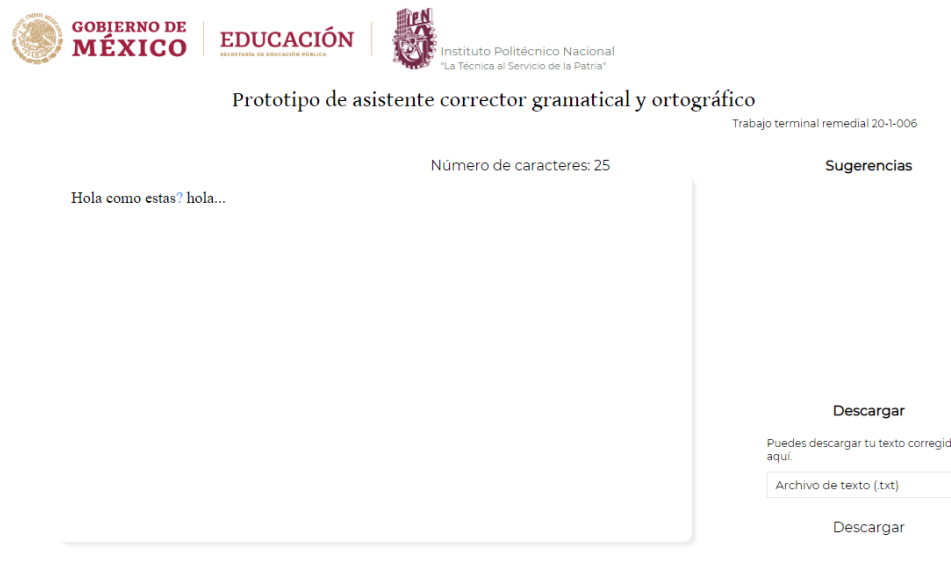


Figura 24 Funcionamiento del sistema, parte 1.

Prototipo de asistente corrector gramatical y ortográfico

Trabajo terminal remedial 20-1-006

Número de caracteres: 298

Este reporte presenta el **de la propuesta** e implementación de un prototipo de asistente corrector gramatical y ortográfico para la redacción de protocolos gramaticalmente correctos, basado en la arquitectura orientada a servicios, denominada software como servicio o **SaaS** por sus siglas en inglés.

Sugerencias

Alerta de gramática

La estructura de tu oración es errónea.

Intenta cambiando la forma en como trasmites tu idea.

Descargar

Puedes descargar tu texto corregido aquí.

Archivo de texto (.txt) ▼

Descargar

Figura 25 Funcionamiento del sistema, parte 2.

Pruebas de regresión

Estas pruebas constan de añadir funciones o soluciones a bugs sin que el sistema caiga o falle. Estas pruebas ayudan a integrar nuevas funciones al sistema ya que nos muestran que no se está recayendo en errores anteriormente solucionados y que las nuevas funciones sean compatibles con la versión actual del sistema.

Al hacer la integración de todos los módulos del sistema, surgieron tres errores:

- Cuando los algoritmos de detección de errores envían la posición del error, en la ventana (front-end) se muestran las palabras de dicho error con un color distinto, sin embargo, para que esto fuera posible se tenían que agregar estilos. Al agregar estos estilos por cada algoritmo, el prototipo no identificaba si eran parte del texto, por lo que al intentar detectar otro error, tomaba en cuenta los estilos añadidos.
 - ✓ Para solucionar esto, se especificó que se debían descartar los estilos del texto al enviarlo a los algoritmos, de tal forma que cuando un texto tuviera diferentes tipos de error, no hubiera problema ni para marcar de otro color las palabras, ni para identificar el texto limpio.
- Al presionar la tecla espacio en el cuadro de texto, todos los espacios que contenía el texto ingresado eran eliminados.
 - ✓ Para evitar esto, en los algoritmos no solo se consideraron las palabras, sino también los espacios.
- Al ingresar texto, se encontraron algunos caracteres que no se reconocían.
 - ✓ Se indicó que el texto ingresado debía tener la codificación UTF-8.

Para el trabajo objetivo de este documento se optaron, además de las pruebas anteriores, realizar pruebas de validación de html y css y pruebas para poner en marcha el servicio. Esto debido a la naturaleza de una aplicación web, en especial, una herramienta tipo SaaS como lo es este prototipo.

Capítulo 5. Implementación

5.1 Resumen

En este capítulo se muestra la forma en que se implementó el prototipo en el servicio de nube seleccionado para su funcionamiento vía Internet.

5.2 Proceso

Para montar el prototipo en la plataforma de Microsoft Azure, fue necesario crear una cuenta de Microsoft. En la figura 21 se muestra la página principal de donde se adquiere la cuenta.

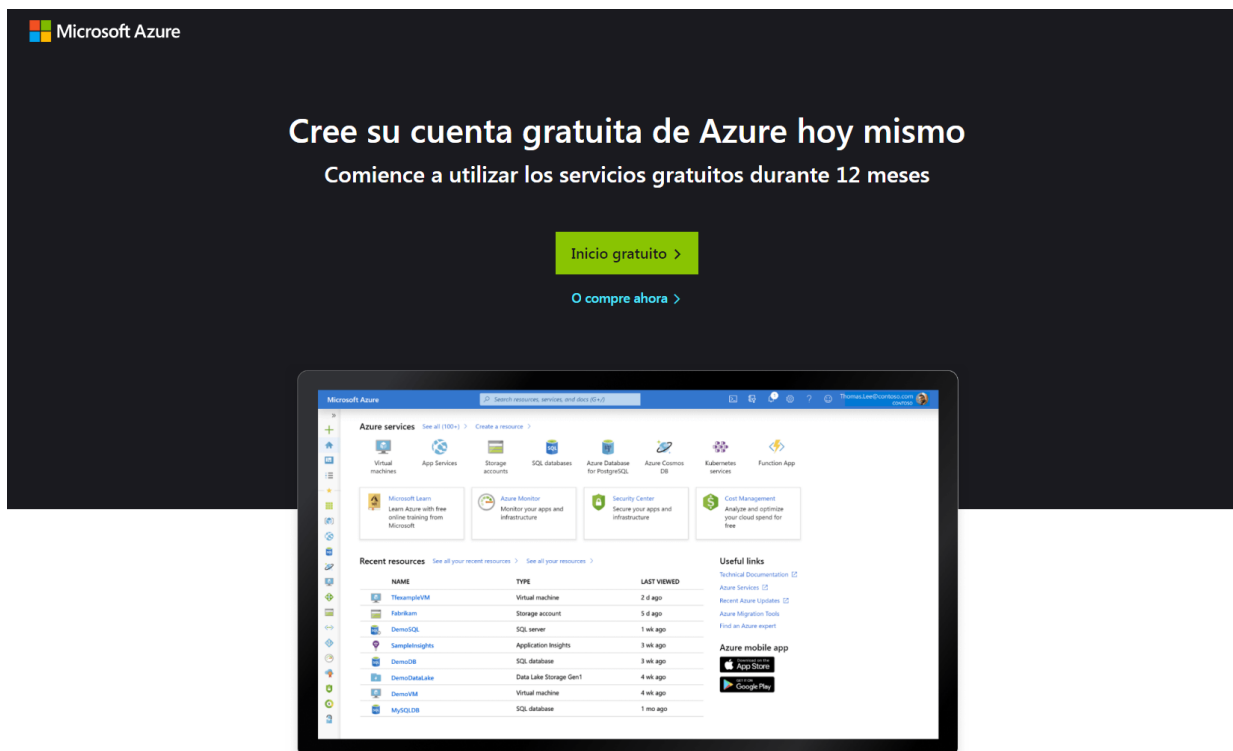


Figura 26 Página de registro de Microsoft Azure

En la creación de la cuenta de Microsoft Azure es gratuita y brinda una cantidad de dinero gratis para gastar en sus propios servicios, la cual está disponible los primeros 30 días. Sin embargo para poder hacer el registro, es necesario contar con una tarjeta de crédito.

Si al hacer uso del servicio no se revasó la cantidad de dinero gratis, no se hacen cargos a la tarjeta, sin embargo, después de los 30 días de prueba, si no se da de baja la cuenta, se empiezan a hacer cargos por los servicios utilizados.

Una vez creada la cuenta, Azure nos da acceso a sus herramientas, permitiendo que el prototipo se suba paso a paso.

La figura 27 muestra la página de inicio o bienvenida de Azure con la barra de herramientas que ofrece.

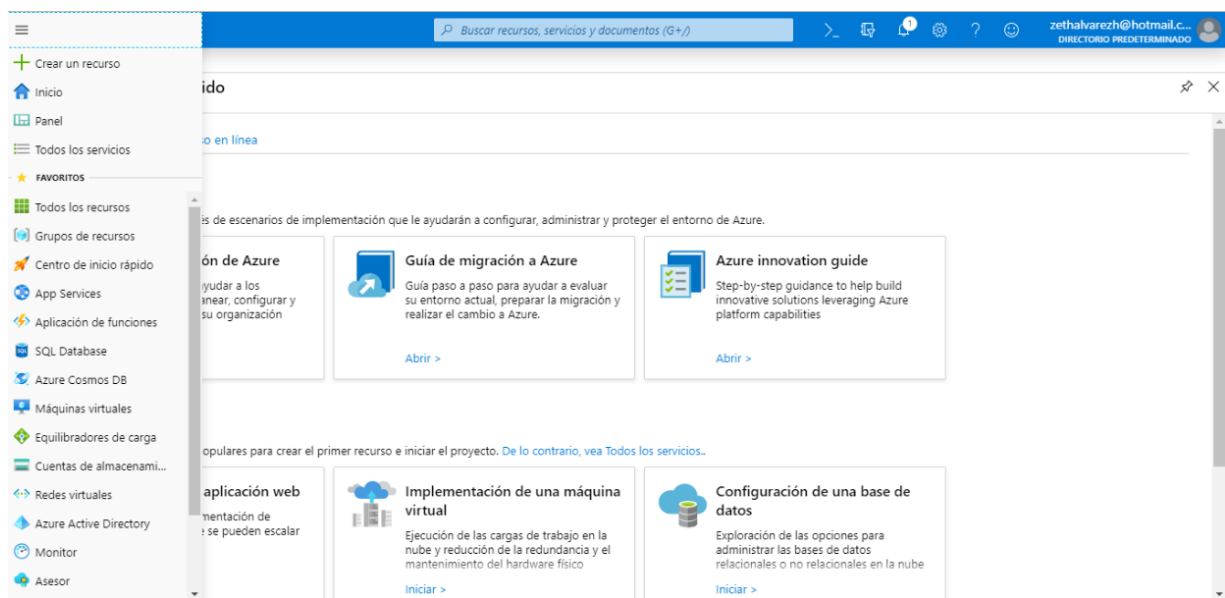


Figura 27 Inicio de Microsoft Azure.

Una vez subido el prototipo, el acceso a éste demora algunos minutos en estar disponible.

Una vez que se tiene acceso, damos por concluido este proceso.

Conclusiones y trabajo a futuro

Conclusiones

Aguirre Hernández Leonardo Miguel:

Este trabajo resultó más complejo de lo que esperábamos, pues con lo desarrollado hasta este punto, pudimos notar que no es una herramienta cien por ciento precisa, dado que únicamente corrige ciertas reglas de la ortografía y de la gramática del español de México. Sin embargo, abre paso a la posibilidad de perfeccionarlo como trabajo a futuro.

Nos permitió cumplir con los objetivos planteados y sin darnos cuenta nos ayudó a mejorar nuestra redacción.

Queda como un trabajo que, a futuro, podría implementarse no únicamente para determinadas personas, sino que podría llevarse incluso a un nivel más general, permitiendo no solo la corrección de protocolos, sino también de otros textos variados.

Álvarez Hernández Zeth:

La complejidad de este trabajo radicó en tratar de implementar las reglas gramaticales y ortográficas en un sistema realice correcciones en textos escolares basadas en dichas reglas.

Debido al léxico variado del idioma español, utilizado en México, un sistema que realice correcciones procurando que se cumplan todas las reglas ortográficas y de gramática resulta difícil de lograr por la cantidad de reglas y el tiempo limitado que tenemos, debido a esto los objetivos que planteamos solo contemplan las reglas ortográficas más sencillas, así como una limitada variedad de reglas gramaticales.

Si bien los objetivos se cumplieron en su mayoría y el prototipo da resultados con un nivel aceptable de exactitud, deja la posibilidad para que en un trabajo a futuro se amplíe la cantidad de reglas y expresiones para mejorar los resultados de las consultas.

Trabajo a futuro

- Añadir más reglas ortográficas y gramaticales.
- Implementar módulo de agregado de nuevas reglas ortográficas o gramaticales.
- Implementar módulo corrector de semántica.
- Implementar módulo de cuentas de usuario.
- Implementar módulo de almacenamiento de archivos

Referencias

- [1] "México, cuarto lugar a nivel mundial en uso de redes sociales", Excélsior, 2018. [En línea]. Disponible en: https://www.excelsior.com.mx/hacker/2018/01/18/1214650?fbclid=IwAR2b4vqV5jS1vF_eG7Fs6c5MwqyJ7kzBqVQR5ZsOHO3Ev3TKE8niA_t5v6I. [Accedido: 08 - Julio - 2019].
- [2] A. Kulkarni and A. Shivananda, Natural Language Processing Recipes. Berkeley, CA: Apress, 2019, p. 2.
- [3] Cf. Berwick y Chomsky, 2011; Berwick, Friederici, Chomsky y Bolhuis, 2013; Chomsky, 1980, 1988.
- [4] Rae.es, 2019. [En línea]. Disponible en: <http://www.rae.es/la-institucion/>. [Consultado el 11 de julio de 2019].
- [5] "Gramática", Rae.es, 2019. [En línea]. Disponible en: <http://www.rae.es/obras-academicas/gramatica>. [Consultado el 11 de julio de 2019].
- [6] R. ASALE, "ortografía", «Diccionario de la lengua española» - Edición del Tricentenario, 2019. [En línea]. Disponible en: <https://dle.rae.es/?id=RG9EvWw>. [Consultado el 11 de julio de 2019].
- [7] Induráin, J.. (2011). *La oración*. En Sintaxis, lengua española (pp. 45-48). Mallorca, 45 - 08029 Barcelona: LAROUSSE.
- [8] Azure.microsoft.com. (2019). Plataforma de inteligencia artificial | Microsoft Azure. [En línea] Disponible en: <https://azure.microsoft.com/es-mx/overview/ai-platform/> [Consultado el 11 de julio de 2019].
- [9] González, L. (2019). Curso Gratis – Introducción a Machine Learning. [Blog] Ligdi González. Disponible en: <http://ligdigonzalez.com/> [Consultado el 11 de julio de 2019].
- [10] Copestake, A. (2004). Natural Language Processing. [En línea] Cl.cam.ac.uk. Disponible at: <https://www.cl.cam.ac.uk/teaching/2002/NatLangProc/revised.pdf> [Consultado el 11 Jul. 2019].
- [11] Diccionario Español de Ingeniería (1.0 edición). Real Academia de Ingeniería de España. 2014. Consultado el 11 de julio de 2019.
- [12] "Documentación de Microsoft Azure", Docs.microsoft.com, 2019. [En línea]. Disponible en: <https://docs.microsoft.com/es-mx/azure/#pivot=get-started&panel=get-started1>. [Consultado el 11 de julio de 2019].
- [13] Executive master in project management, Universidad de Alcalá, "Gestión ágil de proyectos con kanban", 2014 [En línea]. Disponible en: <http://www.uv-mdap.com/programa-desarrollado/bloque-iv-metodologias-agiles/gestion-agil-de-proyectos-con-kanban/> [Accedido: 09 - Julio - 2019].
- [14] A. Stellman and J. Greene, *Learning Agile*, 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., pp. 269 - 368.
- [15] Health Endpoint Monitoring pattern - Cloud Design Patterns", Docs.microsoft.com, 2019. [En línea]. Disponible en: <https://docs.microsoft.com/es-es/azure/architecture/patterns/health-endpoint-monitoring>. [Accedido: 08 - Julio - 2019].
- [16] A. Fox, D. Patterson and S. Joseph, Engineering software as a service. 2a Edición, Strawberry Canyon LLC., 2013, pp. 46 – 53.
- [17] "Salarios de Ingeniero/a en sistemas en Ciudad de México | Indeed.com.mx", Indeed.com.mx, 2019. [Online]. Available: <https://www.indeed.com.mx/salaries/ingeniero-en-sistemas-Salaries,-Ciudad-de-M%C3%A9xico>. [Accessed: 12- Nov- 2019].

Glosario

- **COCOMO**: Constructive Cost Model.
- **CSS**: Hojas de estilo en cascada (Cascading Style Sheets).
- **CU**: Caso de uso.
- **ESCOM**: Escuela Superior de Cómputo.
- **HTML**: Lenguaje de marcado de hipertexto (HyperText Markup Language).
- **IA**: Inteligencia artificial.
- **ML**: Machine learning.
- **PLN**: Procesamiento de lenguaje natural.
- **RAE**: Real Academia Española.
- **RF**: Requisitos funcionales.
- **RN**: Reglas de negocio.
- **RNF**: Requisitos no funcionales.
- **SaaS**: Software como servicio (Software as a Service).
- **SOA**: Service Oriented Architecture.
- **UNAM**: Universidad Nacional Autónoma de México.