



**Instituto Tecnológico y de Estudios Superiores de Monterrey,
Campus Monterrey**

Escuela de Ingeniería y Ciencias

**Inteligencia Artificial Avanzada para la Ciencia de Datos II
Grupo (501)**

Entrega Solución del Reto

Alumnos:

Jorge Chávez Badillo	A01749448
Ariadna Jocelyn Guzmán Jiménez	A01749373
Juan Carlos Varela Téllez	A01367002
Alan Eduardo Aquino Rosas	A01366912
Amy Murakami Tsutsumi	A01750185

Profesores:

Ivan Mauricio Anaya Contreras
Luis Alberto Muñoz Ubando
Blanca Rosa Ruiz Hernandez
Nora Aguirre-Celis
José Eduardo Ferrer
Felipe Castillo Rendón
Jobish Vallikavungal Devassia

Fecha: 4 de diciembre de 2022

Introducción

Problemática

Mediante este proyecto, se buscan resolver problemas con efecto en la industria, sociedad, economía y salud mediante algunos de los objetivos de desarrollo sostenible, los cuales serán una guía importante para que mediante datos del instituto INEGI, podamos analizar cómo mejorar dichas problemáticas e implementar soluciones. De esta forma, lograremos incrementar un impacto social y fortalecer relaciones con distintos grupos de interés. En este caso, nos centraremos únicamente en los siguientes ODS:

- 8: Trabajo decente y crecimiento económico: Busca promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todas y todos.
- 9: Industria, innovación e infraestructura: Pretende conseguir infraestructuras sostenibles, resilientes y de calidad para todos, además de impulsar una industrialización inclusiva para fomentar la innovación.

Para poder abordar los problemas, necesitamos conocer cuáles son las causas por las que se carece de, en el caso de la ODS 8 el trabajo decente y crecimiento económico y por otro lado en la ODS 9, la carencia de innovación e infraestructura.

Teniendo en cuenta ello, lograremos realizar con precisión gráficos visuales para que a través de dictado en texto y voz, se pueda procesar información y realizar una predicción sobre si los datos, coinciden con alguna ODS para poder buscar la implementación de una solución o no.

8 TRABAJO DECENTE
Y CRECIMIENTO ECONÓMICO



9 INDUSTRIA, INNOVACIÓN
E INFRAESTRUCTURAS



Objetivo

Nuestro objetivo es crear un agente que funcione a base del lenguaje natural escrito del usuario. Usando Lenguaje Natural Procesado y modelos de Aprendizaje de Máquina queremos crear un agente que sea capaz de graficar diferentes variables, así como utilizar las gráficas más óptimas que haya para el tipo de variables que se le está pidiendo. De tal forma que estas gráficas se presentan en la aplicación web de forma automática y de una forma estética para que sean de mayor valor para el usuario.

Esto con el fin de que el usuario pueda hacer uso de la información que el INEGI proporciona de una manera más rápida y sencilla.

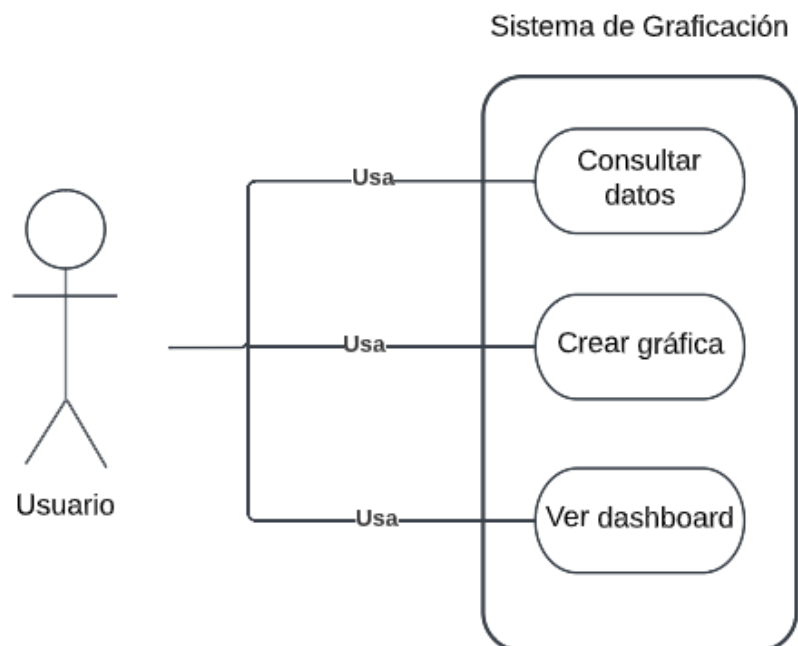
Requerimientos Funcionales

- Se deberán realizar gráficas a través de texto y voz de lenguaje natural.
- Las gráficas que se realicen serán las adecuadas para visualización e interpretación.
- Se debe recopilar información necesaria para la construcción de gráficos desde la base de datos de INEGI
- Utilizar técnicas de NLP (Natural Language Processing) para poder realizar las consultas a la base de datos para finalmente poder realizar las gráficas adecuadas.

Requerimientos No Funcionales

- Los scripts de código necesarios pueden ser programados utilizando diferentes tecnologías o herramientas.
- El sistema debe contar con un manual de usuario para que este pueda ser utilizado correctamente.
- Los tiempos de respuesta para cada una de las funcionalidades deben de ser adecuados y eficientes.
- El sistema debe ser fácil de comprender para que en caso de que existan defectos, estos puedan resolverse fácilmente.
- La arquitectura del software debe ser escalable y modular para futuras iteraciones.
- Se busca limitar o minimizar los errores que puedan presentarse en el sistema.

Diagrama de Casos de Uso



Casos de Uso Extendido

1. Consultar datos.

- Actores: Usuario.
- Objetivo: Consultar la información detallada de la base de datos.
- Contexto: El usuario podrá visualizar en la página web una guía donde podrá encontrar información detallada sobre la organización de la base de datos.

Acción del actor	Respuesta del sistema
1. El usuario ingresa a la página web.	
2. El usuario selecciona la página de inicio.	3. Se despliega una breve descripción sobre el programa y se muestra un pdf como guía de usuario para la consulta de los

	datos.
--	--------

Casos alternativos
2-3 El usuario ignora la interfaz

2. Crear gráfica.

- Actores: Usuario.
- Objetivo: Crear gráfica con las bases de datos del INEGI.
- Contexto: El usuario podrá crear una gráfica con los datos del INEGI que seleccione utilizando el chatbot.

Acción del actor	Respuesta del sistema
1. El usuario ingresa a la página web.	
2. El usuario selecciona la pestaña "Crear".	
3. El usuario ingresa la información que desea graficar.	4. Despliega la gráfica

Casos alternativos
3-4 El usuario ignora la interfaz.

3. Ver dashboard.

- Actores: Usuario.
- Objetivo: El usuario podrá visualizar el dashboard creado con los datos del INEGI.

- Contexto: El usuario podrá visualizar e interactuar con las distintas gráficas.

Acción del actor	Respuesta del sistema
1. El usuario ingresa a la página web.	
2. El usuario selecciona la pestaña "Dashboard".	3. Despliega los gráficos disponibles.
4. El usuario selecciona algún estado para filtrar la información.	5. Despliega los gráficos con el filtro indicado.

Casos alternativos
4-5 El usuario ignora la interfaz.

Herramientas tecnológicas del sistema

- Tecnologías Front End
 - React JS
 - Chart JS
 - Tableau
- Tecnologías Backend
 - Amazon Lambda
 - Amazon S3
 - Jupyter Notebook (Google Colab)
- Tecnologías ML
 - Amazon Lex

Recursos utilizados por el sistema

- Base de datos del INEGI.

Alcance de la Aplicación

La aplicación generará gráficas en la aplicación web, esto significa que el usuario no tendrá la necesidad de moverse de aplicación para poder ver la gráfica que pidió.

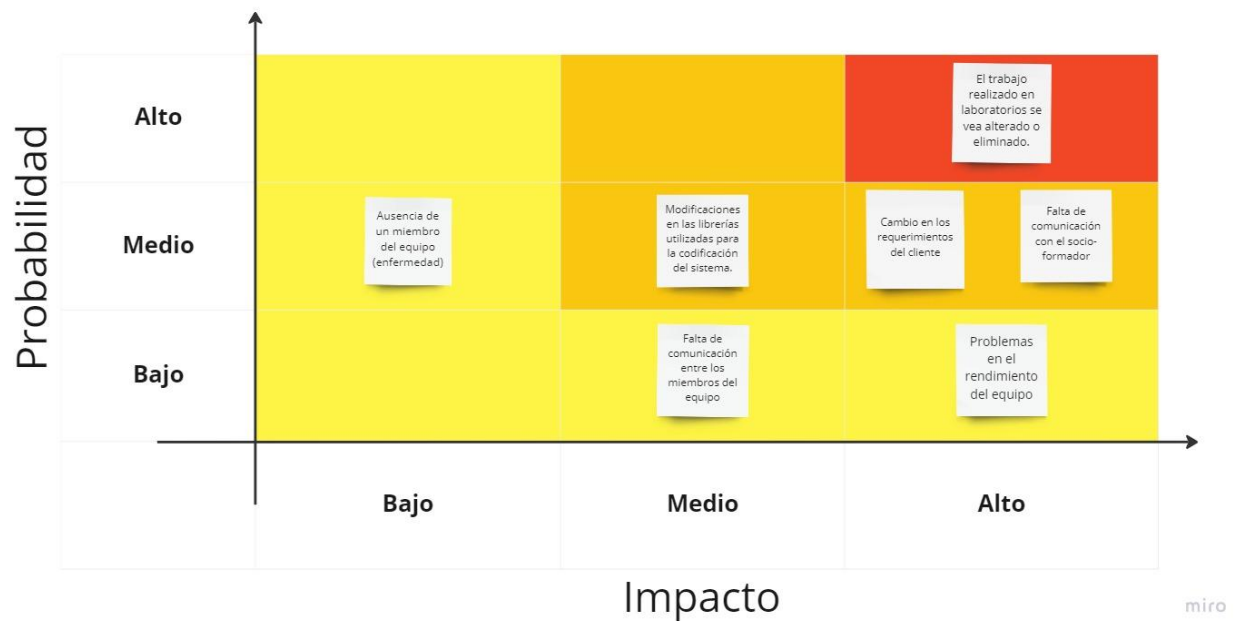
Asimismo, se tiene planeado utilizar Amazon Lex como nuestro receptor del texto natural del usuario ya que cuenta con una implementación relativamente sencilla y con un gran margen de personalización para nuestro problema.

Se utilizará Amazon Lambda para poder controlar el flujo de la información entre nuestros servicios implementados en la nube y la aplicación web.

Por último se utilizará React JS para el desarrollo de las interfaces de usuario, ya que con este framework la implementación es más sencilla y eficiente.

Matriz de Riesgos

[Link a Matriz De Riesgos](#)



Riesgos

1. Ausencia de un miembro del equipo (enfermedad).
2. Falta de comunicación entre los miembros del equipo.
3. Cambio en los requerimientos del cliente.
4. Falta de comunicación con el socio-formador.
5. Problemas en el rendimiento del equipo.
6. Modificaciones en las librerías utilizadas para la codificación del sistema.
7. El trabajo realizado en laboratorios se ve alterado o eliminado.

Plan de mitigación

ID	Descripción de riesgo	Plan de mitigación	Resultado
R1	Ausencia de un miembro del equipo (enfermedad).	El miembro que se ausentará deberá avisar al equipo. Se	Se cumplirá con las actividades en el tiempo establecido.

		llegará a un acuerdo para completar las actividades.	
R2	Falta de comunicación entre los miembros del equipo.	Agendar reuniones para discutir los problemas y utilizar los medios establecidos para tener una comunicación continua.	Mejorar la organización del equipo y cumplir con las actividades de forma colaborativa.
R3	Cambio en los requerimientos del cliente.	Agendar una reunión con el cliente para discutir el cambio del requerimiento. Llegar a un acuerdo para resolver el problema.	Poder realizar los cambios y brindar un producto que cumpla con los requerimientos del cliente.
R4	Falta de comunicación con el socio-formador.	Agendar reuniones periódicas con el socio-formador para compartir los avances y obtener retroalimentación.	Garantizar que el proyecto a implementar satisfice las necesidades del socio-formador
R5	Problemas en el rendimiento del equipo.	Realizar juntas semanales con todos los integrantes del equipo para asegurar que todas las actividades se cumplen de manera	Evitar retrasos y asegurar que no existan problemas en la etapa de implementación.

		correcta y resolver posibles problemas que surgen en la etapa de implementación.	
R6	Modificaciones en las librerías utilizadas para la codificación del sistema.	Definir en el plan inicial de herramientas y/o recursos a utilizar las librerías que se necesitarán para la codificación y realizar una investigación sobre estas.	Al definir las librerías a utilizar en las primeras etapas y al realizar una investigación se evitarán posibles problemas en caso de que se modifiquen.
R7	El trabajo realizado en laboratorios se vea alterado o eliminado.	Realizar copias/backups del trabajo realizado en los laboratorios.	Evitar perder todo el trabajo realizado en los laboratorios.








Metodología

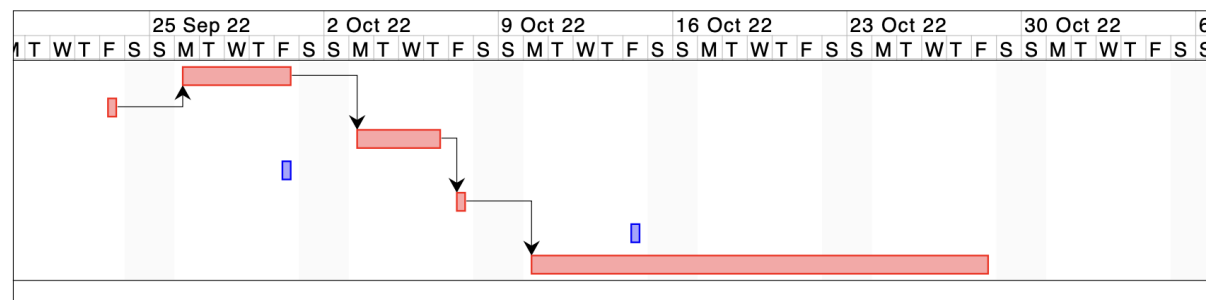
Aunque se trata de un proyecto de Ciencia de Datos, la metodología CRISP-DM no funcionará ya que lo que se busca hacer es una aplicación con servicio que ya incluyen un modelo pre entrenado y no la implementación de un modelo. Es por esto que la metodología que se va a utilizar es la metodología ágil SCRUM.

Esta es una metodología flexible que brilla más en el desarrollo de proyectos.

Se ejecuta en bloques cortos y periódicos llamados *Sprints*. Estos bloques comienzan y acaban en un lapso de 1 a 2 semanas para poder completar el objetivo del *Sprint*. Al terminar cada *Sprint* el equipo habla de los avances, problemas y soluciones que tuvieron, teniendo un énfasis en cómo ser más productivo. Con el pasar del tiempo, el proyecto puede cambiar, pero eso es donde las metodologías ágiles comienzan a diferenciarse de las tradicionales ya que, al ser un proceso iterativo, los cambios de requerimientos se implementan al momento de que acabe un *Sprint*.

Gantt/Cronograma del Reto

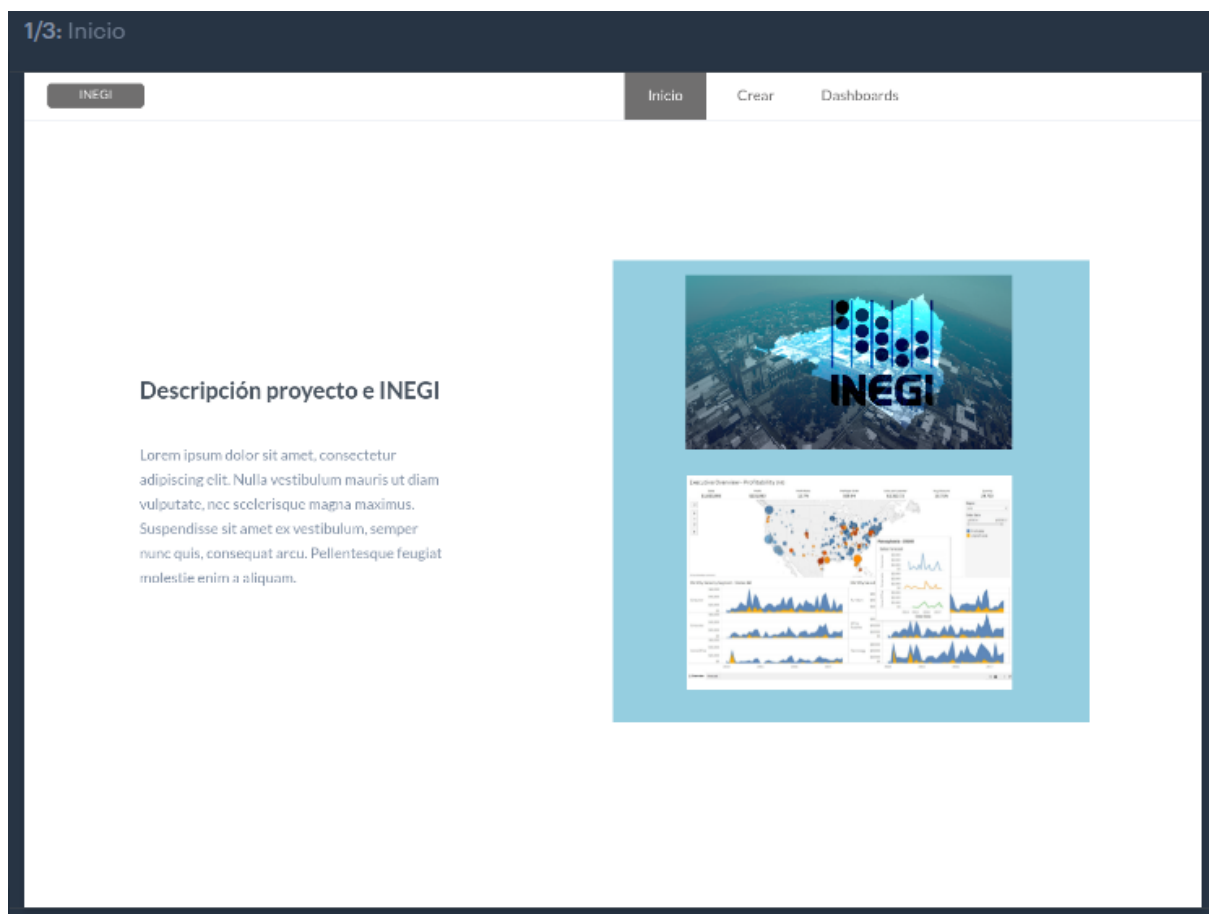
		Name	Duration	Start	Finish	Predecessors
1		Entendimiento del Problema	5 days	9/26/22 8:00 AM	9/30/22 5:00 PM	2
2		Junta 1 Socio Formador	1 day	9/23/22 8:00 AM	9/23/22 5:00 PM	
3		Levantamiento de Requerimientos	4 days	10/3/22 8:00 AM	10/6/22 5:00 PM	1
4		Junta 2 Socio Formador	1 day	9/30/22 8:00 AM	9/30/22 5:00 PM	
5		Junta 3 Socio Formador	1 day	10/7/22 8:00 AM	10/7/22 5:00 PM	3
6		Junta 4 Socio Formador	1 day	10/14/22 8:00 AM	10/14/22 5:00 PM	
7		Implementación de la Primera Fase del Proyecto	15 days	10/10/22 8:00 AM	10/28/22 5:00 PM	5

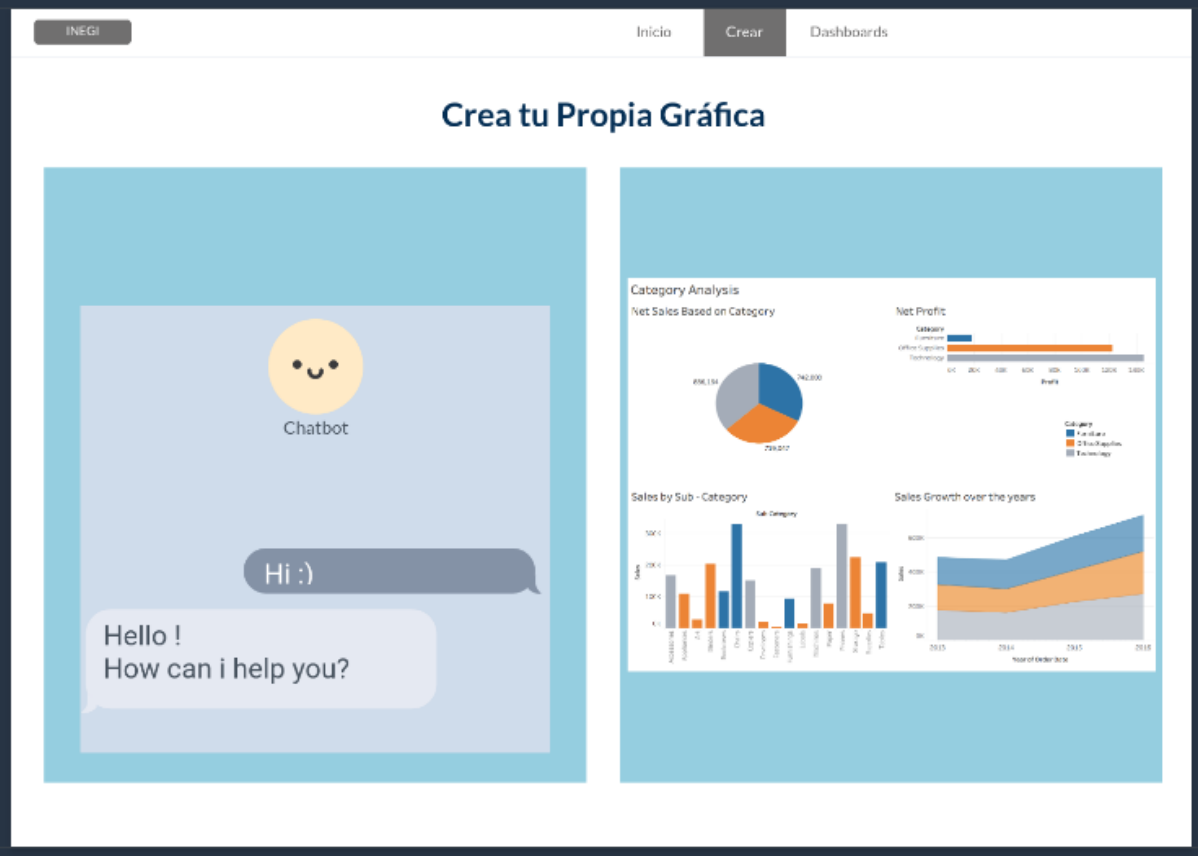


Prototipo

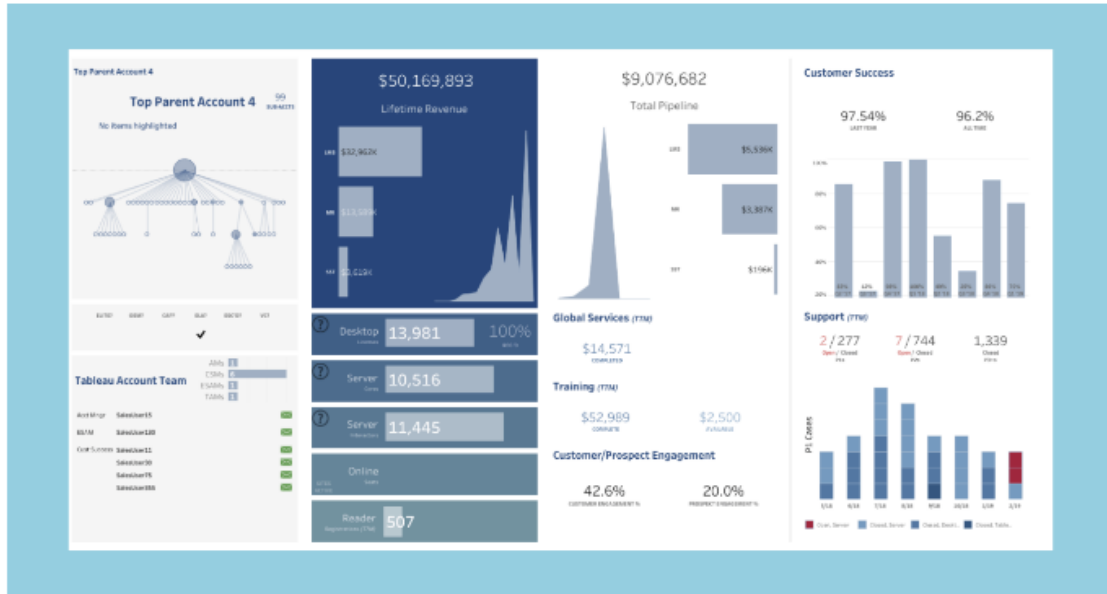
Antes de empezar la codificación de nuestro proyecto, se debe tener una idea clara de lo que queremos lograr y a su vez transmitir y desplegar al usuario, por lo que se hizo un prototipo en la página Marvel para visualizar nuestro objetivo.

Link del prototipo: <https://marvelapp.com/prototype/b7a741h>





Dashboard

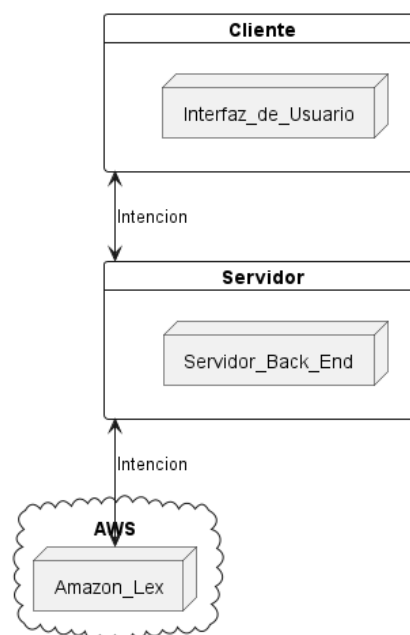


Implementación

Para poder entender la implementación de este proyecto, es necesario primero explicar la arquitectura del mismo, así como explicar de forma considerable el flujo de información entre módulos. Asimismo, se explicará de forma más detallada los servicios y tecnologías que se utilizaron en un apartado más adelante.

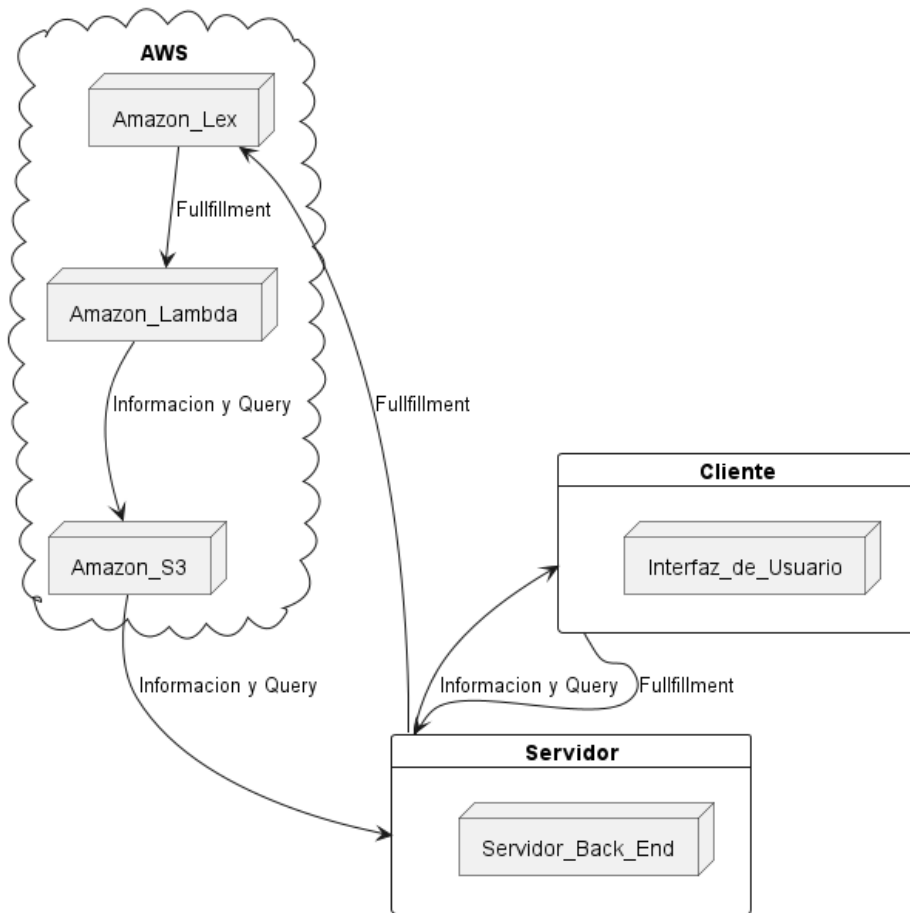
Runtime

Amazon Lex es el servicio que contiene la implementación del agente conversacional, por lo que es encargado de monitorear el flujo de la información con los usuarios. También buscará la intención del usuario para saber qué graficar y de qué manera. Durante toda la conversación, se hacen varias llamadas API con Amazon Lex para que el bot conversacional sepa que responder a la información que proporcione el usuario y ayudarlo a completar su intención.



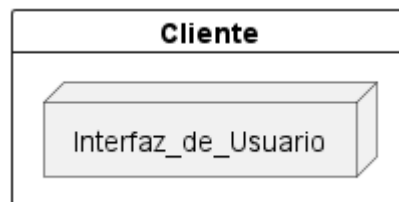
Terminación de la Intención

Una vez que el usuario haya completado su intención, una función en Amazon Lambda se va a activar y va a obtener la información necesaria de Amazon Lex para empezar a graficar. Datos como las columnas que el usuario necesita y el tipo de gráfica van a llegar a esta función para poder hacer el preprocesamiento necesario para pasar al siguiente paso. Después de esto, se crea un JSON y se guarda en Amazon S3 para que el cliente pueda recibir estos datos.



Generación de la Gráfica

Una vez que el cliente obtenga los datos, se utiliza Chart JS para poder generar la gráfica de forma automática. Este paso es el más sencillo en cuestión de conexiones ya que esta operación se hace directamente en el cliente sin necesidad de comunicarse con componentes en el servidor o en la nube.



Adquisición y procesamiento de datos

El INEGI cuenta con una increíblemente vasta base de datos. Es por eso que estos datos no están guardados como simples csv. Estos datos están guardados en archivos llamados parquets.

Los parquets son archivos open source desarrollados por Apache. Estos archivos cuentan con optimización de búsqueda por columna y provee compresión de datos para un mejor rendimiento.

Sin embargo, al ser tipos de archivos específicos, es necesario tener un manejo especial con estos.

Ética de los datos

Debido a que se utilizaron datos del INEGI fue necesario cumplir con algunos elementos sobre el manejo de la información, los cuales se mencionan a continuación.

- **Anonimidad:** El usuario en ningún momento accederá información personal en la aplicación, de ser así, no se guardará esta información. El único componente de la aplicación en el que se debe introducir texto es en el chatbot y este solo funciona con palabras clave acerca de las gráficas o las bases de datos del INEGI.
- **Manejo:** Los datos se obtienen de la base de datos pública del INEGI, por lo que no estamos utilizando datos confidenciales e íntimos. De la misma manera, el usuario no podrá acceder directamente a los datos detallados tales como cantidad de población por manzana o municipio, ya que las gráficas sólo despliegan información general de las estadísticas del censo de 2020.
- **Privacidad:** Dado que no se utilizan o almacenan datos personales del usuario en la aplicación su privacidad no se verá afectada.
- **Autorizaciones:** Dado que las bases de datos utilizadas del INEGI son públicas, se puede decir que contamos con la autorización necesaria brindada por el socio formador para utilizar estos datos, lo cual nos permitió poder desarrollar el proyecto y utilizar dicha información para poder cumplir con todas las funcionalidades.

Apache Spark y PySpark

Apache Spark es un motor multi-lenguaje para analíticas de big data. PySpark es la librería de Python que permite hacer llamadas API directamente con Apache Spark. Estas tecnologías nos permiten adquirir, filtrar, modificar y transformar los datos que se encuentren en un archivo .parquet de una forma eficiente. Debido a la cantidad de datos con los que el INEGI cuenta, los métodos convencionales tardaría varias horas para poder

hacer cualquier tipo de operación simple, es por esto que el uso de Apache Spark y PySpark es necesario.



Después de hacer un proceso, se generaron las siguientes tablas:

- discapacidad.csv
- economia.csv
- educacion.csv
- fecundidad.csv
- hablantes.csv
- migracion.csv
- mortalidad.csv
- poblacion.csv
- religion.csv
- salud.csv
- situacion_conyugal.csv

Haz click [aquí](#) para poder ver con más detalle el proceso y obtención de estos archivos.

Aprendizaje Máquina y Redes Neuronales

Aunque al final hubo una implementación con Amazon Lex, en las primeras semanas de desarrollo se intentó crear un modelo de traducción que tomaba órdenes en lenguaje natural a comandos SQL.

En esta sección se hablará sobre cómo funcionaba este modelo, así como la arquitectura aproximada que utiliza Amazon Lex para la clasificación de Intenciones que provee.

Amazon Lex: Como funciona

AWS es muy discreto en cuánto a sus servicios, como funcionan y lo que pasa detrás de escenas. Es por eso que no hay forma de saber de forma concreta cómo está construido el modelo, las capas que lo conforman y mucho menos sus pesos. Sin embargo, podemos saber un aproximado de cómo está compuesto.

En un [artículo de Amazon Science](#), mencionan el cambio a Redes Neuronales Profundas para el servicio de Alexa presente en varios productos de Amazon. Se habla sobre cómo los científicos de Amazon crearon un pre entrenamiento usando MLM (Masked Language Modeling) no supervisado. Aunque no mencionan ningún tipo de arquitectura para Alexa, mencionan que esta técnica fue inspirada por el modelo BERT para Lenguaje Natural Procesado. Ya que no nos fue posible obtener otra pista de cómo Lex funciona, explicaremos, en la medida de lo posible, como funciona BERT.

BERT

BERT (Bidirectional Encoder Representation from Transformer) es un [artículo](#) que Google publicó en 2018. En este describe un modelo que utiliza Transformer de una forma innovadora, creando resultados considerablemente mejores que otros modelos anteriores.

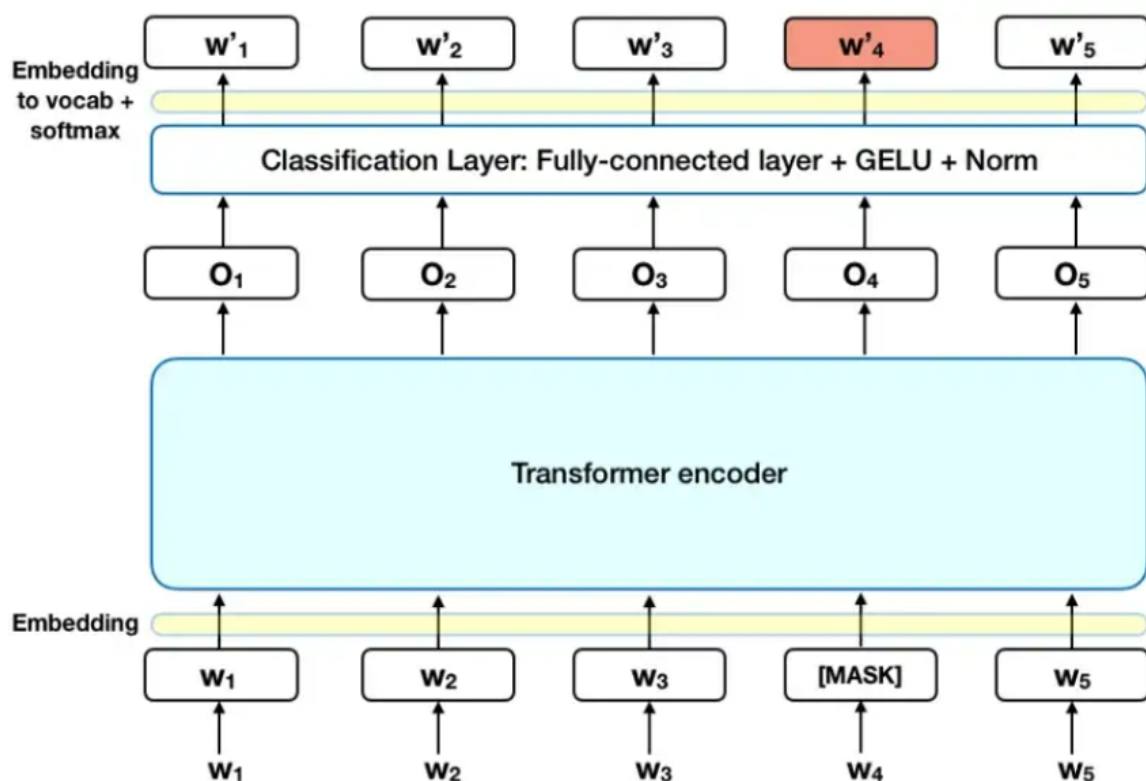
[Transformer](#) es un mecanismo que se basa en atención (habilidad de un modelo en darle más importancia a ciertas palabras conforme la oración se está procesando). En su forma completa, Transformer cuenta con un codificador que lee el input y un decodificador que produce una predicción. Ya que el objetivo de BERT es crear un modelo de lenguaje, solamente el codificador es necesario.

El uso innovador viene justamente del uso del codificador. Normalmente, el input se lee de izquierda a derecha o de derecha a izquierda. A esto se le llama un modelo direccional. Sin embargo, al utilizar el codificador de Transformer ya que este codificador permite la lectura completa de la oración al mismo tiempo. Esto es importante ya que esto permite al modelo poder aprender el contexto de cada palabra tomando en cuenta las palabras restantes, anteriores y posteriores de la oración.

Sin embargo, el uso de un modelo bidireccional limita el entrenamiento ya que no se les puede poner un objetivo claro. Modelos direccionales predicen la siguiente palabra, o la palabra anterior, algo que modelos como BERT no pueden hacer sin un poco de ayuda:

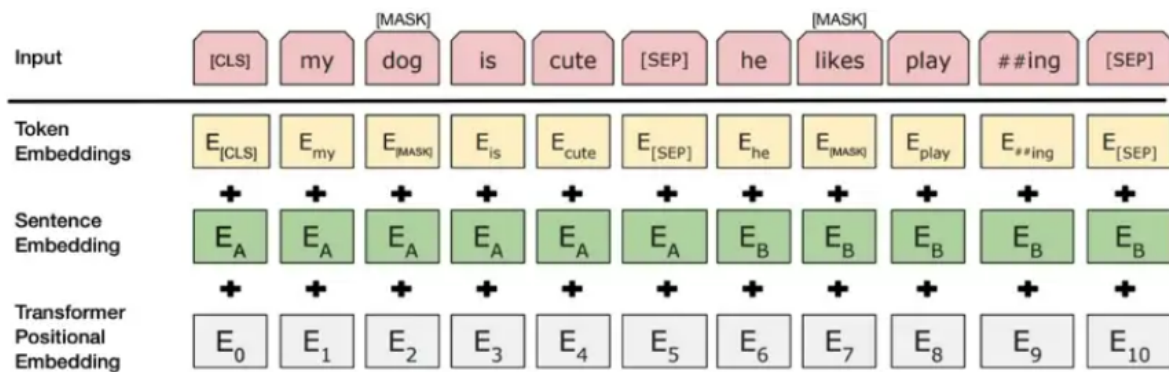
Masked Language Modeling (MLM)

Antes de dar oraciones para entrenar, alrededor de 15% de las palabras son reemplazadas por el token `[MASK]`. El modelo luego intenta predecir el valor original del token basado solamente en el contexto que las demás palabras en la oración proveen. Para que esto funcione, es necesaria la implementación de una capa de clasificación, que prediga la mejor palabra de todo el vocabulario que tiene el modelo.



Next Sentence Prediction (NSP)

Además de MLM, BERT también utiliza NSP en su proceso de entrenamiento. Se le presentan 2 oraciones y el modelo intenta predecir si la segunda oración está relacionada con la primera y, por consecuencia, es subsecuente a esta. El 50% de los pares de oraciones consiste de oraciones que son subsecuentes, mientras que el otro 50% tiene oraciones que no tienen que ver una con la otra.



Cuando BERT se está entrenando, tanto MLM como NSP son procesados al mismo tiempo para minimizar la función de pérdida combinada entre las 2 estrategias.

Dos años después de que el artículo de BERT fuera publicado en 2018, Google lo implementaría directamente en su motor de búsqueda, mostrando la gran capacidad y cambio de estigma para los modelos de NLP.

Modelo de traducción: Speech-2-SQL

Como ya se había mencionado antes, en los primeros días de desarrollo, se intentó hacer un modelo que tomara lenguaje natural a comandos SQL. Este modelo tenía como objetivo independizarnos de servicios de terceros para poder tener control total del desarrollo del proyecto, sin embargo, el gran tiempo de entrenamientos (incluso con optimización de GPU), la falta de datos para entrenar (incluso con data augmentation), la complejidad del problema y problemas de tiempo, fue necesaria la implementación de Amazon Lex. Sin embargo, nuestros esfuerzos no fueron en vano ya que se aprendió sobre los modelos de traducción, así como el aumento de datos y NLP en general. En esta sección se mostrará nuestro modelo, aunque no se haya implementado y no esté completo.

Tecnologías utilizadas

La obtención de datos, el entrenamiento del modelo y la inferencia del mismo fueron desarrollados en un Jupyter Notebook con un kernel Python 3.10. El framework que se utilizó para la creación del modelo fue TensorFlow.



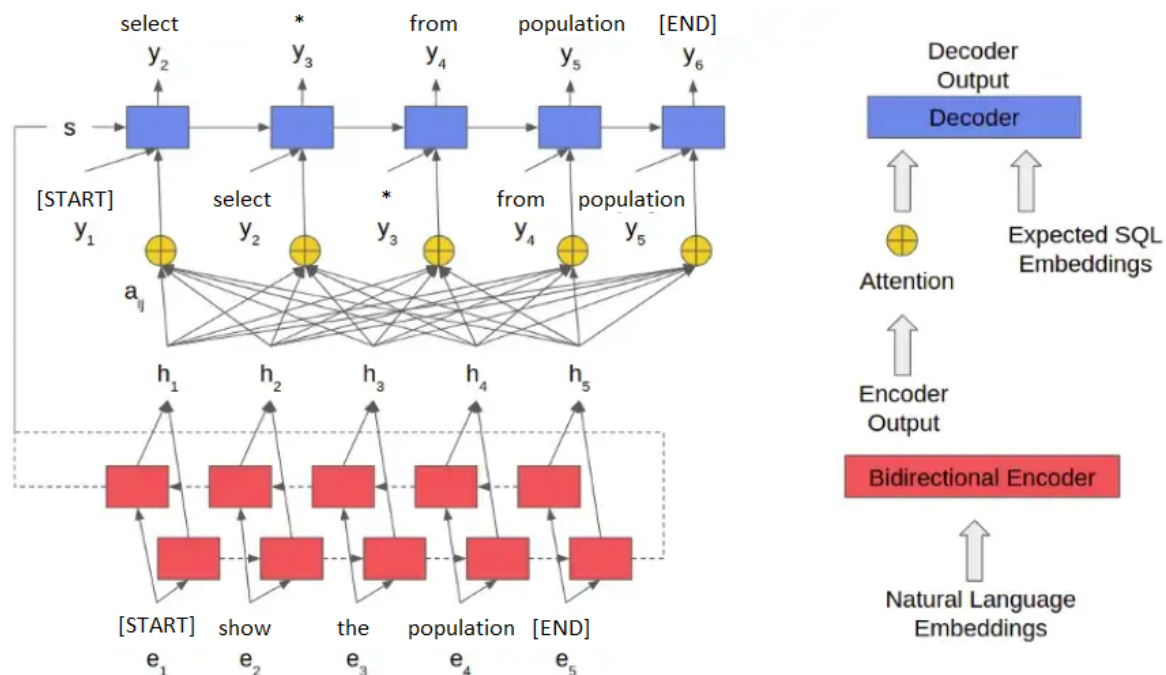
Arquitectura y tipo de problema

La red neuronal fue una RNN. Las redes neuronales recurrentes se utilizan mucho en el ámbito del lenguaje natural procesado debido a que, a comparación de inferencia numérica donde se busca que los datos sean independientes, el input de cualquier problema de lenguaje natural procesado son oraciones cuyas palabras dependen de la palabra anterior e incluso de la palabra siguiente. Es por esto que las RNN son muy utilizadas para este tipo de problemas.

Asimismo, el modelo es un modelo sequence-to-sequence. En esencia, nuestra problemática es una de traducción, y este tipo de modelos seq-2-seq consiste en entrenar modelos que tomen un input de un dominio (lenguaje natural) a otro (dialecto SQL).

Nuestro texto es insertado en un codificador bidireccional, pero a diferencia de BERT, este modelo cuenta con un decodificador, cuyo input es el estado final del codificador. Antes de entrar al decodificador, nuestro input pasa por una capa de atención. Después de pasar por esta capa de atención, se combina con las palabras incrustadas de SQL así como palabras

incrustadas que el modelo puede ir aprendiendo. Esto es porque es posible que nuestro modelo encuentre palabras que no estuvieron al momento de tokenizar nuestros



datos. El decodificador al final nos da nuestro comando en SQL.

Preprocesamiento y obtención de datos

Sin duda esta fue la parte más complicada al momento de crear este modelo. La creación del dataset tuvo que ser local ya que no hay una base de datos que tenga esta problemática específica en mente. Para crear los datos, fue necesario hacer una lista de variaciones de cómo el usuario podría pedir las columnas, también con sus respectivos comandos SQL. Asimismo, debido a la gran cantidad de tablas y columnas, fue necesario crear una variación de cada variación, al igual que su respectivo comando SQL. No solamente eso, queríamos que nuestro modelo pudiera entender ciertas condiciones, como ordenar de forma ascendente o descendente, así como dar condicionales *mayor que*, *menor que* o *igual*, y que estas condiciones también se refirieron a otras columnas en un mismo comando. También por esto se tuvo que hacer más variaciones de las variaciones. Y por último, no esperamos que nuestro usuario escriba el comando tal cual como nosotros lo escribimos, sino que esperamos una ligera variación, como sinónimos o diferentes formas de decir las cosas. Es aquí donde se utilizó una base de datos de paráfrasis.

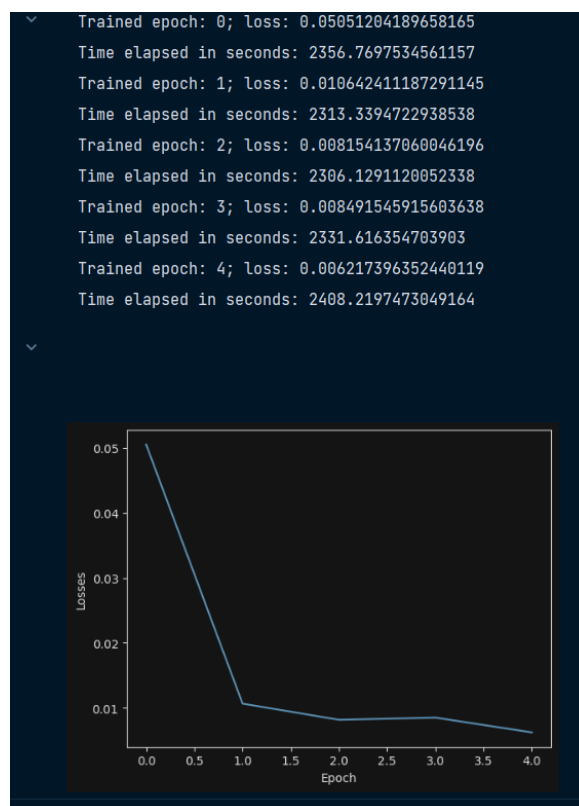
Es así como 6 variantes de un comando combinado con 50 columnas diferentes aproximadamente se convirtieron en una vasta base de datos de 520 MB.

Entrenamiento

Debido a la gran cantidad de datos y a la pobre implementación del código de entrenamiento, tuvimos que limitarnos a 5 epoch de entrenamiento ya que cada epoch tardaba aproximadamente 40 minutos en completarse. Asimismo, se utilizó gradiente descendente como nuestro algoritmo de optimización.

Resultado

Al final, el rendimiento de nuestro modelo fue... Cuestionable. Aunque nuestra función de pérdida parecía estar muy cercana a 0, al momento de probar el modelo, no salió ningún tipo de comando SQL aceptable, ni utilizando las oraciones más básicas.



```
1 print('show me the general population in ascending order')
2 print(translate('show me the general population in ascending order', model))

✓ show me the general population in ascending order
(1, 10, 128)
('select from situacionconyugal asc', <tf.Tensor: shape=(5, 1, 1, 10), dtype=float32, numpy=
array([[[[1.44861382e-03, 6.12490578e-04, 7.42245684e-05,
          4.48834291e-03, 4.31410503e-04, 6.90730230e-05,
          1.20093151e-04, 1.97905847e-05, 4.11297689e-04,
          9.92324710e-01]]],
        [[2.41557844e-02, 4.98008309e-03, 7.00054824e-01,
```

Debido a la complejidad de la problemática y los problemas de tiempo ya que se planeaba hacer, no solo un modelo, sino toda una aplicación, se decidió dejar este modelo y reemplazarlo por lo que se convertiría en nuestro bot conversacional con Amazon Lex. Sin embargo, este modelo de traducción nos enseñó bastantes cosas en el mundo de NLP que sin duda utilizaremos en el futuro.

Para ver a más detalle la implementación de este modelo, puede hacer click [aquí](#).

Servicios y tecnologías utilizadas

Ya que se tiene clara la arquitectura de la aplicación, así como los modelos que están detrás de los servicios que ocupamos, vamos a explicar de forma detallada los servicios que se utilizan, como funcionan y las diferentes implementaciones que se desarrollaron.

Componentes en la Nube

Como ya se había mencionado, hay diferentes servicios en la nube que se utilizan para que la aplicación web funcione de forma correcta.

Estos servicios los proporciona Amazon Web Services, una plataforma en la nube que cuenta con decenas de servicios.

Cabe mencionar que se ocupó AWS, debido a que diferencia de un hosting local con sus propios servidores dedicados, los proveedores de servicios en la nube ofrecen un esquema flexible donde se se paga unicamente por lo que se ocupa .Esta es una excelente opción para proyectos de este tipo donde no se tiene el presupuesto, ni la infraestructura para levantar los servicios de la aplicación localmente.

En esta sección vamos a describir qué hacen estos servicios que se utilizaron, así cómo explicar las implementaciones.



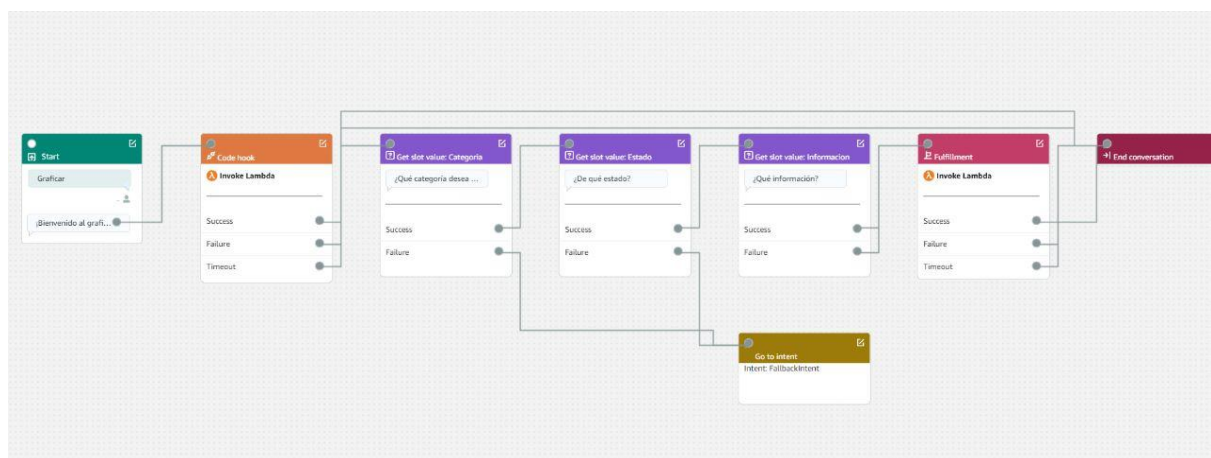
Amazon Lex

Probablemente el servicio más importante que se utiliza en la aplicación web. Amazon Lex es un servicio para construir interfaces conversacionales para aplicaciones con el uso de voz o texto. Este servicio nos permite tener la flexibilidad de tener diferentes intenciones en nuestro chatbot, así como una implementación completa.



Mediante la creación de bot en Amazon Lex, se generó un *intent* que reconoce los datos ingresados del usuario para poder realizar una gráfica en base a las cifras del INEGI. En él, se agregaron los slots: categoría, estado e información; datos importantes que necesitamos del usuario para poder realizar las consultas en la base de datos y desplegar en la interfaz el esquema solicitado por el usuario.

Nuestro bot, es de los servicios más importantes que utilizamos para la implementación del reto, ya que es la base de la comunicación entre sistema y usuario



Flujo conversacional de agente "Amazon Lex"

Draft version

Spanish (LATAM)

Successfully built

Build

Test

Sample utterances (10)

Info

Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

Filter

Sort by added (ascending)

Preview

Plain Text

Graficar

Quiero graficar

Me permites graficar

Me gustaria graficar

{Categoria}

Me interesa grafica

Quiero graficar la categoria de {Categoria}

{Categoria} de {Estado}

Editor

Visual builder

New

Save intent

Declaración de ejemplos de intención

▼ Slots (3) - optional

Info

Information that a bot needs to fulfil the intent. The bot prompts for slots required for intent fulfilment, in priority order below.

Add slot

Filter

▶ Prompt for slot: Categoria

Message: ¿Qué categoría desea graficar?

Slot type

categoria

×

▶ Prompt for slot: Estado

Message: ¿De qué estado?

Slot type

estado

×

▶ Prompt for slot: Informacion

Message: ¿Qué información?

Slot type

informacion

×

Declaración de Slots que ocupará el agente conversacional

< Slot types (3)

Sort by last updated ▼

informacion

categoria

estado

Slot type values

Modify the list of values used to train the machine-learning model to recognise values for a slot.

POB1

Población total X

Poblacion total X

X

Total de personas X

Cantidad de personas X

Tab or ; for a new value

POB31

Total de mujeres X

Cantidad de mujeres X

X

Población total femenina X

Poblacion total femenina X

Mujeres X

Tab or ; for a new value

POB57

Población total masculina X

Poblacion total masculina X

Cantidad de hombres X

Hombres X

Homvres X

Onvres X

X

Value

Tab or ; for a new value

Add value

Declaración de SlotsTypes y sinónimos que ocupará el agente conversacional

< Slot types (3)

Sort by last updated ▼

informacion

categoria

estado

Slot type values

Modify the list of values used to train the machine-learning model to recognise values for a slot.

poblacion

habitantes X

gente X

X

cantidad de personas X

numero de personas X

población X

Tab or ; for a new value

fecundidad

nacimientos X

nacieron X

nacidos X

X

natalidad X

Tab or ; for a new value

mortalidad

Tab or ; for a new value

X

migracion

migraron X

Tab or ; for a new value

X

hablantes

lengua indigena X

lengua X

X

Tab or ; for a new value

Value

Tab or ; for a new value

Add value

Declaración de SlotsTypes y sinónimos que ocupará el agente conversacional

Slot types (3)

Sort by last updated ▼

informacion

categoria

estado

Slot value resolution

☐ Expand values (default)
Values used as training data.

☒ Restrict to slot values
Use only values provided.

Slot type values

Mexico	<div>México X</div> <div>Mejico X</div> <div>Mex X</div> <div>Mex. X</div> <div>Méx X</div> <div>Méx. X</div> <div>Tab or ; for a new value</div>
Jalisco	<div>Xalisco X</div> <div>Guadalajara X</div> <div>Jal X</div> <div>Jal. X</div> <div>Tab or ; for a new value</div>
Veracruz	<div>Beracruz X</div> <div>Veracruz X</div> <div>Beracruz X</div> <div>Ver. X</div> <div>Ver X</div> <div>Tab or ; for a new value</div>
Puebla	<div>Pue. X</div> <div>Pue X</div> <div>Puebla X</div>

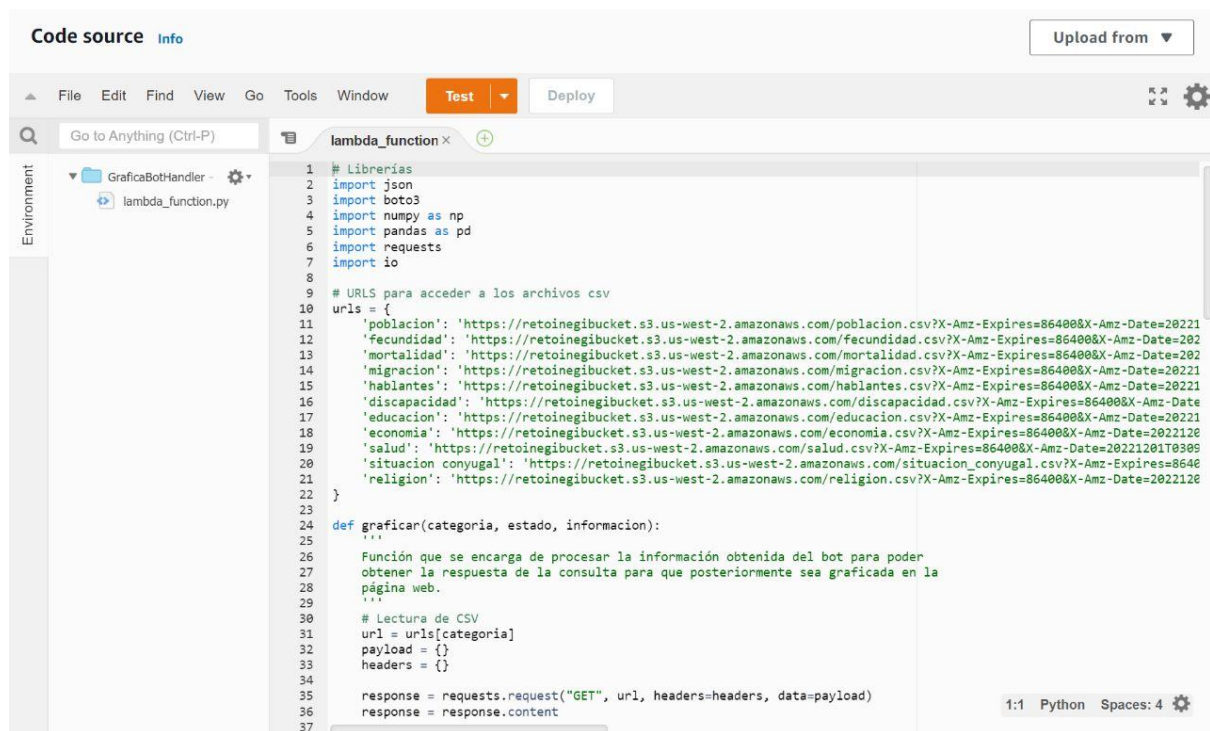
Save slot

Declaración de SlotsTypes y sinónimos que ocupará el agente conversacional

Amazon Lambda

Lambda es un servicio que nos permite ejecutar código en base a eventos, sin necesidad de servidores. Esto significa que no hay necesidad de configurar un servidor, manejar un sistema operativo o hacerse cargo de los recursos de la instancia que lo está ejecutando. Con este servicio nos podemos conectar a cualquier servicio que tenga AWS.

El servicio de lambda, fue de los más importantes en la implementación del reto, ya que nos ayudó a poder realizar la codificación necesaria para consultar nuestros archivos csv almacenados en Amazon S3 y conocer cuál era el adecuado para obtener la información solicitada por el usuario. Éste código, tendría como resultado un archivo txt en formato json que contendría únicamente los datos correspondientes a la categoría, estado e información donde posteriormente, serían leídos por ReactJS (una de nuestras tecnologías locales) para poder desplegar los datos en el gráfico requerido.



```
1 # Librerías
2 import json
3 import boto3
4 import numpy as np
5 import pandas as pd
6 import requests
7 import io
8
9 # URLs para acceder a los archivos csv
10 urls = {
11     'poblacion': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/poblacion.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
12     'fecundidad': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/fecundidad.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
13     'mortalidad': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/mortalidad.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
14     'migracion': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/migracion.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
15     'hablantes': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/hablantes.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
16     'discapacidad': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/discapacidad.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
17     'educacion': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/educacion.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
18     'economia': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/economia.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
19     'salud': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/salud.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
20     'situacion conyugal': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/situacion_conyugal.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309',
21     'religion': 'https://retoinegibucket.s3.us-west-2.amazonaws.com/religion.csv?X-Amz-Expires=86400&X-Amz-Date=20221120T0309'
22 }
23
24 def graficar(categoria, estado, informacion):
25     """
26     Función que se encarga de procesar la información obtenida del bot para poder
27     obtener la respuesta de la consulta para que posteriormente sea graficada en la
28     página web.
29     """
30     # Lectura de CSV
31     url = urls[categoria]
32     payload = {}
33     headers = {}
34
35     response = requests.request("GET", url, headers=headers, data=payload)
36     response = response.content
37
```

Amazon Simple Storage System (Amazon S3)

Amazon S3 es un servicio para el almacenamiento de archivos en la nube. Nos permite guardar cualquier tipo de información, a diferencia de una base de datos que permite texto plano, en un sistema de almacenamiento para después poder utilizarlo en aplicaciones.



Mediante nuestro bucket **retodatainegibucket**, se almacenaron los archivos de datos csv correspondientes a cada categoría, además, se almacenaron de misma forma el archivo **bot_response.txt**, que contenía los datos para la gráfica solicitada por el usuario en un formato json para después ser leído por el cliente.

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

Access analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

Dashboards

AWS Organizations settings

Feature spotlight 3

► AWS Marketplace for S3

retodatainegibucket info

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (12)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

↻

Copy S3 URI

Copy URL

Download

Open

Delete

Actions ▼

Create folder

Upload

Find objects by prefix

☐

Name

▲

Type ▼

Last modified ▼

Size ▼

Storage class ▼

☐

bot_response.txt

txt

December 2, 2022, 12:43:20 (UTC-06:00)

58.0 B

Standard

☐

discapacidad.csv

csv

December 2, 2022, 11:59:43 (UTC-06:00)

3.9 KB

Standard

☐

economia.csv

csv

December 2, 2022, 11:59:44 (UTC-06:00)

3.2 KB

Standard

☐

educacion.csv

csv

December 2, 2022, 11:59:45 (UTC-06:00)

3.2 KB

Standard

☐

fecundidad.csv

csv

December 2, 2022, 11:59:45 (UTC-06:00)

2.2 KB

Standard

☐

hablantes.csv

csv

December 2, 2022, 11:59:47 (UTC-06:00)

1.4 KB

Standard

☐

migracion.csv

csv

December 2, 2022, 11:59:48 (UTC-06:00)

2.4 KB

Standard

☐

mortalidad.csv

csv

December 2, 2022, 11:59:50 (UTC-06:00)

987.0 B

Standard

☐

poblacion.csv

csv

December 2, 2022, 11:59:42 (UTC-06:00)

3.8 KB

Standard

☐

religion.csv

csv

December 2, 2022, 11:59:53 (UTC-06:00)

1.5 KB

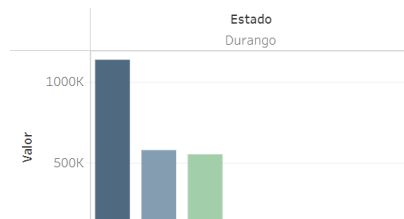
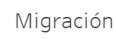
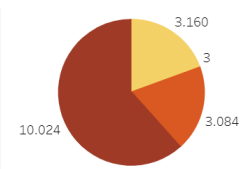
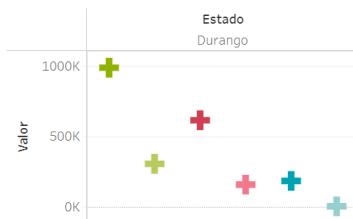
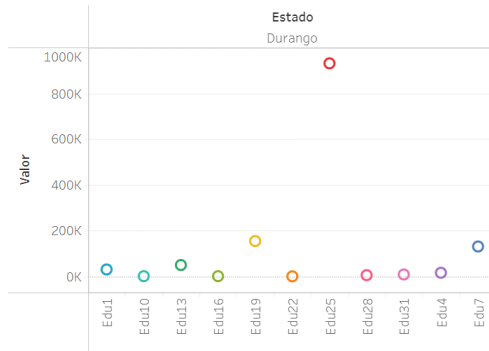
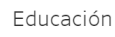
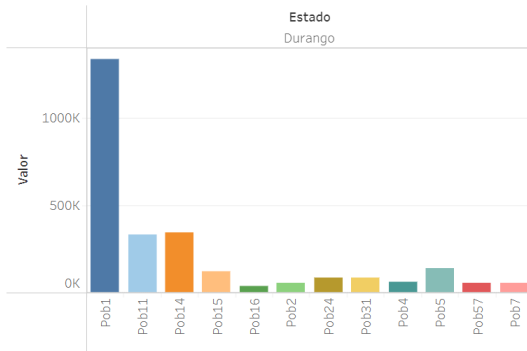
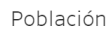
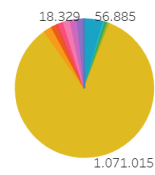
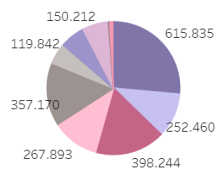
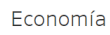
Standard

Tableau Public

Por último, Tableau Public. Esta es una plataforma que permite subir dashboards hechos con el software Tableau. Esta plataforma nos permite también incrustar estos dashboards a páginas web para su uso e interacción. Este último servicio no está vinculado a AWS en ninguna forma.



Tableau fue la plataforma utilizada para poder desplegar a nuestros usuarios un dashboard, donde se encontrará información gráfica de manera dinámica importante de cada categoría de acuerdo al estado que se seleccionara.



Tecnologías locales

Además de utilizar servicios en la nube, también se utilizan tecnologías locales para el buen funcionamiento de la aplicación web.

React

React js es un framework front end de javascript para hacer interfaces web responsivas.



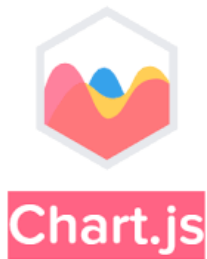
ReactJS fue el framework que nos ayudó para poder realizar la página web donde el usuario tendría la interacción con el bot y los datos del INEGI. En él se realizaron componentes como: Inicio, Crear y Dashboard, los cuales contenían la codificación necesaria para poder desplegar cada uno de estos apartados.

```

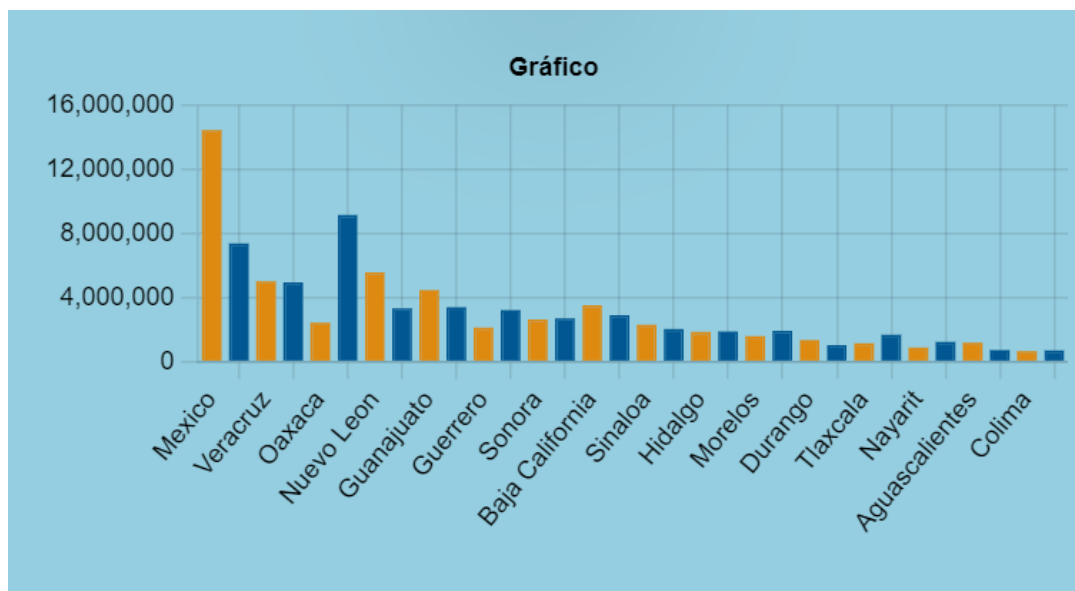
reto-inegi-front > src > components > Crear.js > Crear
1  /*
2  Nombre del archivo: Crear.js
3  Fecha de creación: 28-10-2022
4
5  Página que permitirá al usuario crear gráficas con un dataset de
6  entrada y a través de un chatbot en Amazon Lex.
7  */
8
9  import "../styles/Crear.css";
10 import AWS from "aws-sdk";
11 import React, { useEffect, useState } from "react";
12 import AOS from "aos";
13 import "aos/dist/aos.css";
14 import { Bar, Doughnut, Pie } from "react-chartjs-2";
15 import Chart from "chart.js/auto";
16
17 // Define a color
18 Chart.defaults.color = "#000000";
19
20 const Crear = () => {
21   useEffect(() => {
22     AOS.init();
23     AOS.refresh();
24   }, []);
25
26   const [result, setResult] = useState(null);
27   const [grafica, setGrafica] = useState({
28     labels: [],
29     datasets: [
30       {
31         data: [
32           10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790, 800, 810, 820, 830, 840, 850, 860, 870, 880, 890, 900, 910, 920, 930, 940, 950, 960, 970, 980, 990, 1000, 1010, 1020, 1030, 1040, 1050, 1060, 1070, 1080, 1090, 1100, 1110, 1120, 1130, 1140, 1150, 1160, 1170, 1180, 1190, 1200, 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290, 1300, 1310, 1320, 1330, 1340, 1350, 1360, 1370, 1380, 1390, 1400, 1410, 1420, 1430, 1440, 1450, 1460, 1470, 1480, 1490, 1500, 1510, 1520, 1530, 1540, 1550, 1560, 1570, 1580, 1590, 1600, 1610, 1620, 1630, 1640, 1650, 1660, 1670, 1680, 1690, 1700, 1710, 1720, 1730, 1740, 1750, 1760, 1770, 1780, 1790, 1800, 1810, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030, 2040, 2050, 2060, 2070, 2080, 2090, 2100, 2110, 2120, 2130, 2140, 2150, 2160, 2170, 2180, 2190, 2200, 2210, 2220, 2230, 2240, 2250, 2260, 2270, 2280, 2290, 2300, 2310, 2320, 2330, 2340, 2350, 2360, 2370, 2380, 2390, 2400, 2410, 2420, 2430, 2440, 2450, 2460, 2470, 2480, 2490, 2500, 2510, 2520, 2530, 2540, 2550, 2560, 2570, 2580, 2590, 2600, 2610, 2620, 2630, 2640, 2650, 2660, 2670, 2680, 2690, 2700, 2710, 2720, 2730, 2740, 2750, 2760, 2770, 2780, 2790, 2800, 2810, 2820, 2830, 2840, 2850, 2860, 2870, 2880, 2890, 2900, 2910, 2920, 2930, 2940, 2950, 2960, 2970, 2980, 2990, 3000, 3010, 3020, 3030, 3040, 3050, 3060, 3070, 3080, 3090, 3100, 3110, 3120, 3130, 3140, 3150, 3160, 3170, 3180, 3190, 3200, 3210, 3220, 3230, 3240, 3250, 3260, 3270, 3280, 3290, 3300, 3310, 3320, 3330, 3340, 3350, 3360, 3370, 3380, 3390, 3400, 3410, 3420, 3430, 3440, 3450, 3460, 3470, 3480, 3490, 3500, 3510, 3520, 3530, 3540, 3550, 3560, 3570, 3580, 3590, 3600, 3610, 3620, 3630, 3640, 3650, 3660, 3670, 3680, 3690, 3700, 3710, 3720, 3730, 3740, 3750, 3760, 3770, 3780, 3790, 3800, 3810, 3820, 3830, 3840, 3850, 3860, 3870, 3880, 3890, 3900, 3910, 3920, 3930, 3940, 3950, 3960, 3970, 3980, 3990, 4000, 4010, 4020, 4030, 4040, 4050, 4060, 4070, 4080, 4090, 4100, 4110, 4120, 4130, 4140, 4150, 4160, 4170, 4180, 4190, 4200, 4210, 4220, 4230, 4240, 4250, 4260, 4270, 4280, 4290, 4300, 4310, 4320, 4330, 4340, 4350, 4360, 4370, 4380, 4390, 4400, 4410, 4420, 4430, 4440, 4450, 4460, 4470, 4480, 4490, 4500, 4510, 4520, 4530, 4540, 4550, 4560, 4570, 4580, 4590, 4600, 4610, 4620, 4630, 4640, 4650, 4660, 4670, 4680, 4690, 4700, 4710, 4720, 4730, 4740, 4750, 4760, 4770, 4780, 4790, 4800, 4810, 4820, 4830, 4840, 4850, 4860, 4870, 4880, 4890, 4900, 4910, 4920, 4930, 4940, 4950, 4960, 4970, 4980, 4990, 5000, 5010, 5020, 5030, 5040, 5050, 5060, 5070, 5080, 5090, 5100, 5110, 5120, 5130, 5140, 5150, 5160, 5170, 5180, 5190, 5200, 5210, 5220, 5230, 5240, 5250, 5260, 5270, 5280, 5290, 5300, 5310, 5320, 5330, 5340, 5350, 5360, 5370, 5380, 5390, 5400, 5410, 5420, 5430, 5440, 5450, 5460, 5470, 5480, 5490, 5500, 5510, 5520, 5530, 5540, 5550, 5560, 5570, 5580, 5590, 5600, 5610, 5620, 5630, 5640, 5650, 5660, 5670, 5680, 5690, 5700, 5710, 5720, 5730, 5740, 5750, 5760, 5770, 5780, 5790, 5800, 5810, 5820, 5830, 5840, 5850, 5860, 5870, 5880, 5890, 5900, 5910, 5920, 5930, 5940, 5950, 5960, 5970, 5980, 5990, 6000, 6010, 6020, 6030, 6040, 6050, 6060, 6070, 6080, 6090, 6100, 6110, 6120, 6130, 6140, 6150, 6160, 6170, 6180, 6190, 6200, 6210, 6220, 6230, 6240, 6250, 6260, 6270, 6280, 6290, 6300, 6310, 6320, 6330, 6340, 6350, 6360, 6370, 6380, 6390, 6400, 6410, 6420, 6430, 6440, 6450, 6460, 6470, 
```

Chart JS

Biblioteca de JavaScript de código abierto para la visualización de datos.



Dentro de nuestra implementación, ya procesados los datos que solicita el cliente mediante el agente conversacional, estos se regresan al cliente en un archivo .txt en formato json, para después ser graficados con los métodos de componentes que proporciona Chart.js.

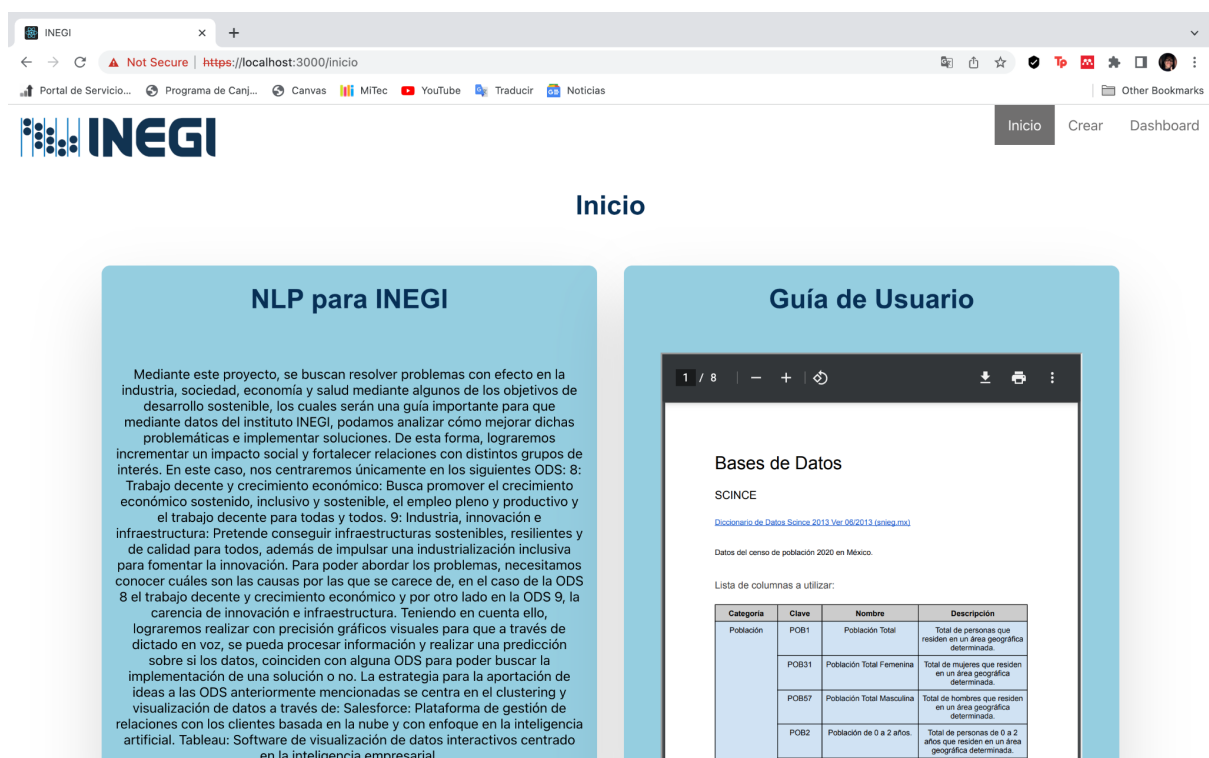


Resultado Final

Combinando todas estas tecnologías y servicios, podemos observar la aplicación resultante, la cual cuenta con tres principales secciones.

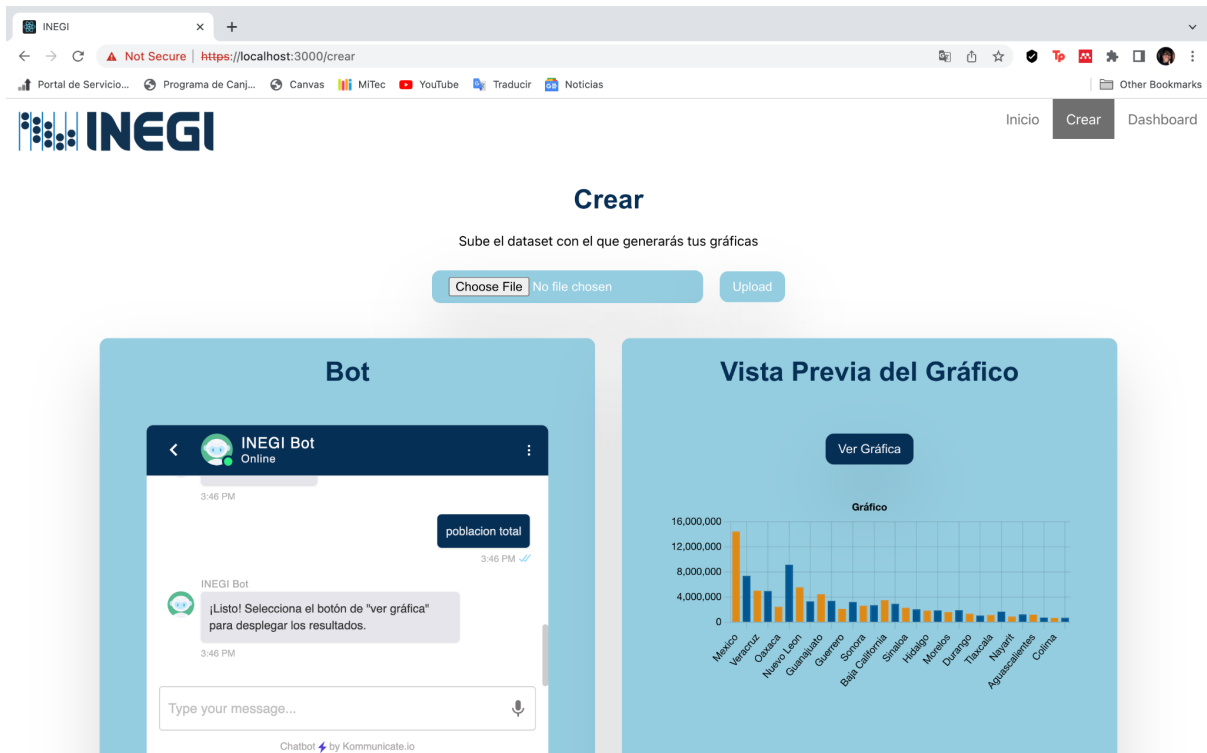
Inicio

Página que dará una descripción al usuario sobre la funcionalidad de la interfaz y sus objetivos. Además, se mostrará una guía de usuario detallada sobre la información que puede ser consultada en la base de datos del INEGI.



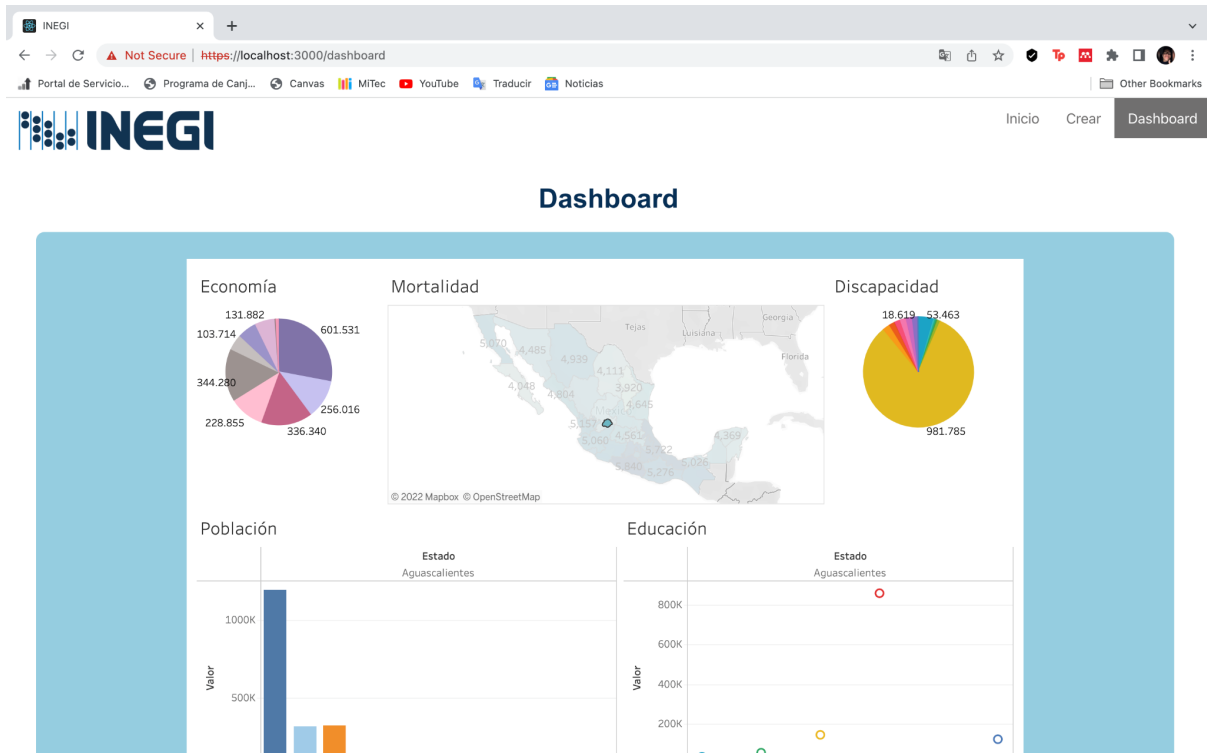
Crear

Página que permitirá al usuario crear gráficos con un dataset de entrada y a través de un chatbot en Amazon Lex. La ventana del lado izquierdo incluye un bot que recibirá comandos naturales. Una vez que el usuario termine de dar su instrucción, la gráfica resultante de esta instrucción se podrá observar en la segunda parte.



Dashboard

Página que mostrará el resultado de un análisis que se hizo a la base de datos que nuestro socio formador INEGI nos proporcionó. Contiene diferentes gráficas que muestran diferentes estadísticas de la población en general.



Futuras iteraciones/Cosas que mejorar

Para futuras iteraciones del proyecto, se espera que pueda funcionar para cualquier formato de dataset que esté disponible. Aunque en el momento, todavía no funciona completamente con cualquier tipo de dataset debido a los diferentes tipos de representaciones que puede haber los datos y el reto que significa interpretar correctamente esto por el bot, ya dentro de la misma implementación del proyecto existe código funcional que es la base para crear un agente conversacional que se adapte a las necesidades de cada dataset, sin importar la representación de los datos que se suba. Con cada iteración y versión de la aplicación se agrega compatibilidad con más fuentes de datos, haciendo la aplicación más flexible a las necesidades del usuario.

Link a Repositorio

Link a la organización: <https://github.com/retoINEGI>

Front End: <https://github.com/retoINEGI/front-end>

Back End: <https://github.com/retoINEGI/back-end>

Documentación: <https://github.com/retoINEGI/documentacion>

Link a vídeos de evidencia final

Vídeo explicativo: [Evidencia Final de Proyecto Equipo 7 - Text/Voice to Graph INEGI - YouTube](#)

Vídeo demostrativo:

<https://drive.google.com/file/d/1F9K9JJvv2vvkxTYZoBcShuhP3wlqtQNh/view?usp=sharing>

Conclusión

A lo largo de la concentración de inteligencia artificial avanzada para la ciencia de datos nos fue posible aprender sobre las diferentes áreas de la inteligencia artificial, así como las herramientas que la complementan y consideramos que cada uno de los módulos que fueron impartidos a lo largo del semestre fueron de gran ayuda puesto que el conocimiento adquirido de alguna manera fue implementado en la solución del reto, de igual manera, el haber generado una aplicación web tan completa, donde la integración de diferentes tecnologías es un punto clave para el buen funcionamiento de la misma, toda esta integración de herramientas tecnológicas fueron un gran reto para el equipo, pues fue necesario pensar en una arquitectura que fuera totalmente funcional, escalable y segura, que garantizara que la aplicación web final cumpliera plenamente con los requerimientos planteados.

En cuanto a la desarrollo y aplicación de Deep learning, nos llevamos 2 lecciones clave. Al momento de realizar nuestro modelo experimentamos de primera mano dos de las limitantes que creemos desfavorecen el desarrollo más rápido de una comunidad de practicantes en estas áreas .

La primera limitante, es la gran cantidad de información que se necesita para que la aplicación de NLP funcione correctamente, pues incluso con la generación de 2 millones de datos sintéticos con data augmentation y manualmente, no le bastó al modelo para llegar a un rango aceptable en base a métricas en el validation y test set para nuestra particular aplicación, considerando que se verificó que el modelo no tuviera algún tipo de underfitting.

Ahora en cuanto la segunda limitante, es los requerimientos de hardware para correr estos algoritmos con aplicaciones más complejas. Debido a que no contábamos con una gran cantidad de gpus para acelerar las operaciones matriciales debíamos correr un modelo y en caso de no funcionar, modificar la arquitectura y empezar desde cero. En el caso de tener varias gpu se podrían entrenar los modelos en paralelo en lugar de casarnos con un mismo modelo, sin embargo, esto se encuentra fuera de nuestro alcance y creemos que la mayoría de practicantes nuevos se encuentran en la misma situación.

Sin embargo, consideramos que estas limitantes, igualmente nos forzaron a investigar sobre arquitecturas más eficientes como bert y su aplicación para NLP, y creemos que el conocimiento obtenido respecto a la creatividad con las que se diseñan estas arquitecturas nos pueden servir como influencia y inspiración para crear nuevas arquitecturas y compartirlas con la comunidad.

Finalmente, hablando de la aplicación en general, podemos concluir que la aplicación desarrollada es bastante completa y cumple con lo requerido, sin embargo, consideramos algunos elementos que pueden ser trabajo a futuro para una posible mejora a la aplicación web en general, algunas de estas mejoras serían la implementación de más tipos de gráficas tanto para la parte realizada con chart.js como la parte del dashboard de tableau, otro elemento importante es la implementación del modelo de deep learning para una predicción más inteligente de acuerdo al data source necesitado dentro de nuestra solución, entre algunas otras que podrían enriquecer a el proyecto en un futuro.