**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)**

**Momento de Retroalimentación: Reto Documentación**

**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 noviembre de 2022

# Requerimientos

## Problemática

La accesibilidad en el mundo tecnológico es una rama muy importante para todos, pero al mismo tiempo se habla muy poco de esta. Hacer que la tecnología y los diferentes programas estén al alcance de todos es necesario.
Es por esto que nuestro programa objetivo en este proyecto es Tableau, un software diseñado para la fácil interpretación de grandes volúmenes de datos y que es utilizado por toda la industria.

## Objetivo

Nuestro objetivo es crear un agente que funcione a base de la voz 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. Estas gráficas se presentan en Tableau como si hubieran sido creadas con la interfaz gráfica del mismo programa.

## Requerimientos Funcionales

- Se deberán realizar gráficas a través de voz.
- 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 o cualquier base de datos.
- Utilizar técnicas de NLP (Natural Language Processing) para poder realizar las consultas a la base de datos para finalmente poder realizar las gráficas adecuadas.

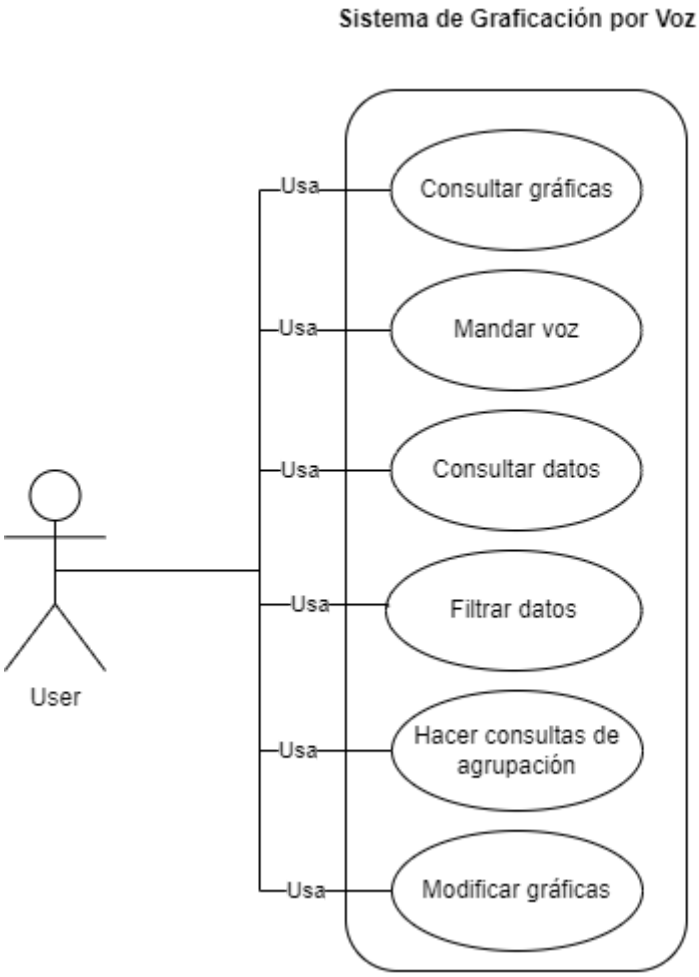
## Requerimientos No Funcionales

- 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.

- Se busca limitar o minimizar los errores que puedan presentarse en el sistema.

## Diagrama de Casos de Uso

[Link al diagrama]




## Casos de Uso Extendido

1. Consultar gráficas.
    - Actores: Usuario.
    - Objetivo: El usuario podrá visualizar gráficas en base a los datos de INEGI.
    - Contexto: El usuario podrá visualizar las gráficas que se han realizado previamente utilizando los datos de INEGI.

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. El usuario ingresa a la página web. | |

| | |
|---|---|
| 2. El usuario selecciona la opción “Consultar gráficas”. | 3. Despliega los gráficos disponibles. |

| Casos alternativos |
|---|
| 2-3 No hay historial de gráficos |

2. Mandar voz.
    - Actores: Usuario.
    - Objetivo: Se realizan gráficos en Tableau mediante una grabación de voz.
    - Contexto: Se necesitan los audios del usuario para poder interpretar datos y mandarlos a Tableau para la construcción de una gráfica recomendada.

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. Ingresa a la página web. | |
| 2. El usuario selecciona la opción “Grabar audio”.<br>3. El usuario brinda los permisos al sistema para el uso del micrófono. | 4. Convierte la información a texto.<br>5. Procesa e interpreta la información en Python.<br>6. Se ejecutan consultas SQL.<br>7. Manda la información a Tableau.<br>8. Despliega el gráfico. |

| Casos alternativos |
|---|
| 3-4 El usuario no brinda correctamente los permisos al sistema.<br>3-4 No se detecta correctamente el audio del usuario |

3. Consultar datos.
    - Actores: Usuario.
    - Objetivo: El usuario podrá visualizar los datos ya almacenados en el sistema.
    - Contexto: Se creará un historial de los datos ingresados al sistema sobre la INEGI para que el usuario pueda consultarlos en cualquier momento.

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. El usuario ingresa a la página web. | |
| 2. El usuario selecciona la opción “Consultar datos”. | 3. Despliega los datos disponibles. |

| Casos alternativos |
|---|
| 2-3 No hay historial de datos |

4. Filtrar datos.
    - Actores: Usuario.
    - Objetivo: Filtrar los datos existentes para lograr un resultado gráfico deseado.
    - Contexto: El usuario podrá interactuar con los datos para examinar e interpretar gráficas en Tableau.

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. El usuario ingresa a la página web. | |
| 2. El usuario selecciona la opción “Consultar gráficas”. | 3. Despliega los gráficos disponibles. |
| 4. El usuario manipula los datos a través de la interfaz incrustada de Tableau. | |

| Casos alternativos |
|---|
| 4 El usuario ignora la interfaz. |

5. Hacer consultas de agrupación.
    - Actores: Usuario.
    - Objetivo: El usuario podrá realizar un recuento sobre algún dato en particular.
    - Contexto: El usuario podrá manipular los datos y observar operaciones como promedios, conteos, máximos, mínimos, entre otros.

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. El usuario ingresa a la página web. | |
| 2. El usuario selecciona la opción “Consultar gráficas”. | 3. Despliega los gráficos disponibles. |
| 4. El usuario manipula los filtros a través de la interfaz incrustada de Tableau.<br>5. El usuario selecciona el dato de agrupación deseado. | |

| Casos alternativos |
|---|
| 4 El usuario ignora la interfaz. |

6. Modificar gráficas.
    - Actores: Usuario.
    - Objetivo: El usuario podrá cambiar el diseño de las gráficas.
    - Contexto: El usuario interactúa con la interfaz de Tableau para poder realizar modificaciones a los gráficos existentes.

| Acción del actor | Respuesta del sistema |
|---|---|
| 6. El usuario ingresa a la página web. | |
| 7. El usuario selecciona la opción “Consultar gráficas”. | 8. Despliega los gráficos disponibles. |
| 9. El usuario manipula los gráficos a través de la interfaz incrustada de Tableau. | |

| Casos alternativos |
|---|
| 4 El usuario ignora la interfaz. |

## Herramientas del Sistema

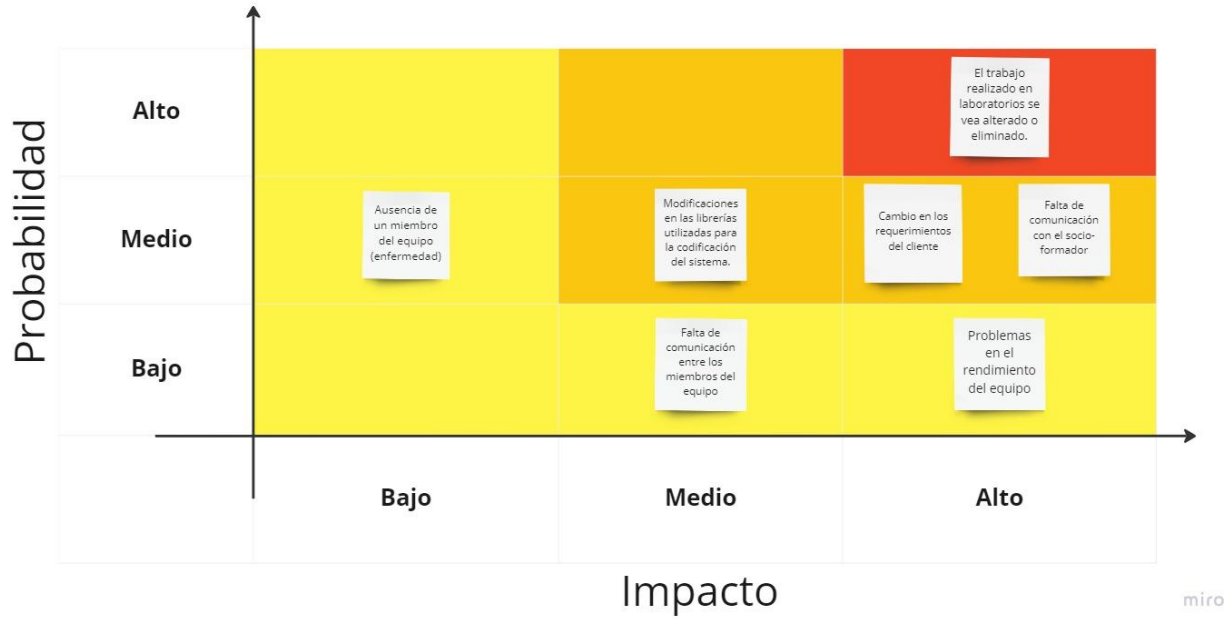
- Base de datos del INEGI.
- Tableau
- Salesforce
- Google Colab / Jupyter Notebook
- Amazon Transcript

## Alcance de la Aplicación

La aplicación generará gráficas en Tableau, esto significa que la aplicación no hará las gráficas, sino que utilizara un API que conecte directamente para la generación de worksheets con Tableau.
Asimismo, se tiene planeado utilizar AWS Amazon Transcribe para la generación de texto desde el habla para facilitar el procesamiento de lenguaje natural a estrictamente texto.
Por último se utilizará Neuraan, un API que limpia la entrada de texto con NLP para poder crear los querys de Tableau de una mejor forma. Por lo tanto, el desarrollo de la aplicación se enfocará en la traducción de las palabras del usuario a las gráficas brindadas por Tableau.

## 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

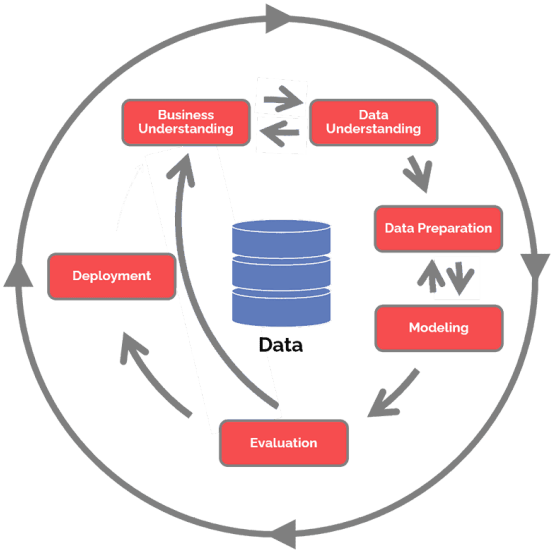
| ID | Descripción de riesgo | Plan de mitigación | Resultado |
|---|---|---|---|
| R1 | Ausencia de un miembro del equipo (enfermedad). | El miembro que se ausentará deberá avisar al equipo. Se llegará a un acuerdo para completar las actividades. | Se cumplirá con las actividades en el tiempo establecido. |
| 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 satisface las necesidades del socio-formador |
| R5 | Problemas en el rendimiento del equipo. | Realizar juntas semanales con todos los | Evitar retrasos y asegurar que no existan problemas |

| | | | |
|---|---|---|---|
| | | integrantes del equipo para asegurar que todas las actividades se cumplen de manera correcta y resolver posibles problemas que surgen en la etapa de implementación. | 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

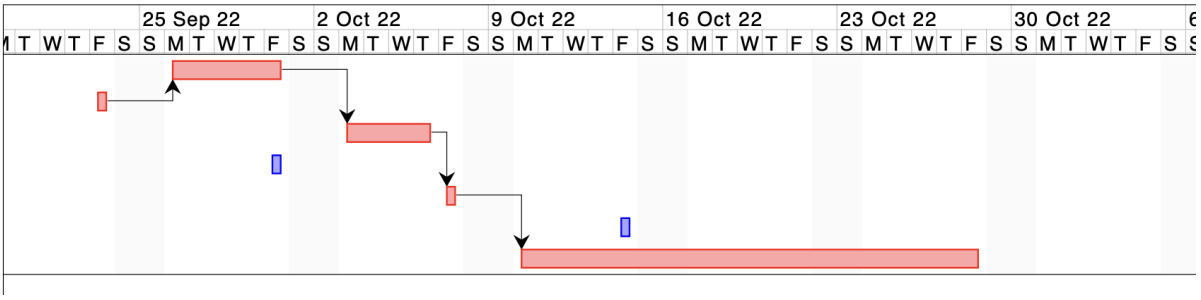
La metodología que se va a utilizar es la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). Esta metodología es estándar en los proyectos de Ciencia de Datos y Aprendizaje Máquina. Esta metodología cuenta con 6 pasos principales:

- Entendimiento de la empresa: Esto se refiere al entendimiento de lo que la empresa necesita. En otras palabras, el problema que la empresa necesita solucionar.
- Entendimiento de los datos: Esto se refiere a los datos que necesitamos, los datos que vamos a buscar y la composición de dichos datos (si están limpios, necesitan transformarse, etc)
- Preparación de los datos: En caso de que los datos sean inconsistentes, se hará la limpieza y transformaciones necesarias para el correcto procesamiento de estos.
- Modelación: cuando ya se tengan los datos limpios, se comenzará a diseñar un modelo que se apegue al objetivo del proyecto.
- Evaluación: se medirá el rendimiento del modelo y si el rendimiento es aceptable, se podrá pasar al siguiente paso.
- Despliegue: se implementará el modelo en el ambiente real.




## 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 |




# Diseño

## Prototipo

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

*Link del prototipo:* https://marvelapp.com/prototype/b7a741h

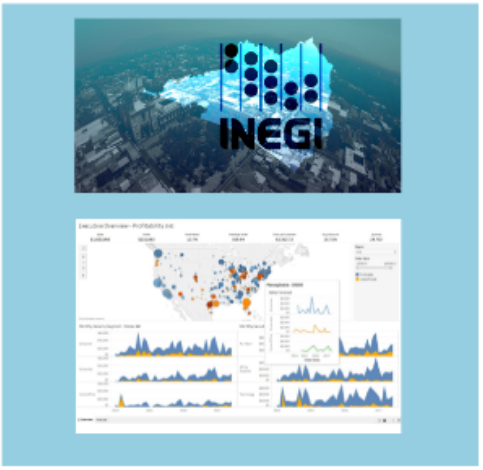
**1/3:** Inicio

INEGI

Inicio Crear Dashboards

## Descripción proyecto e INEGI

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla vestibulum mauris ut diam vulputate, nec scelerisque magna maximus. Suspendisse sit amet ex vestibulum, semper nunc quis, consequat arcu. Pellentesque feugiat molestie enim a aliquam.

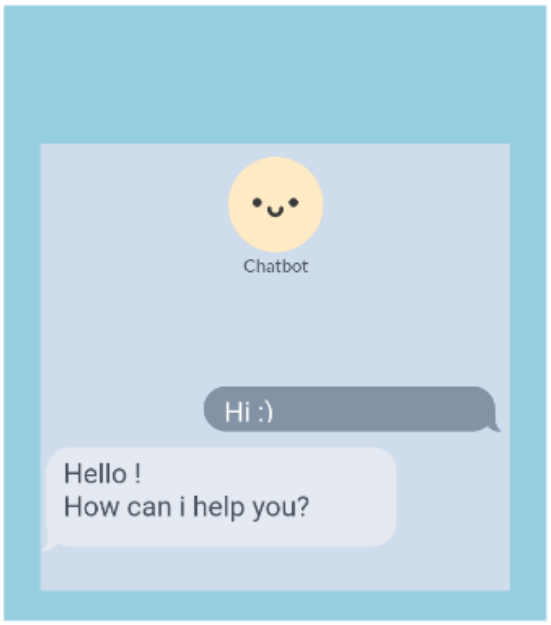


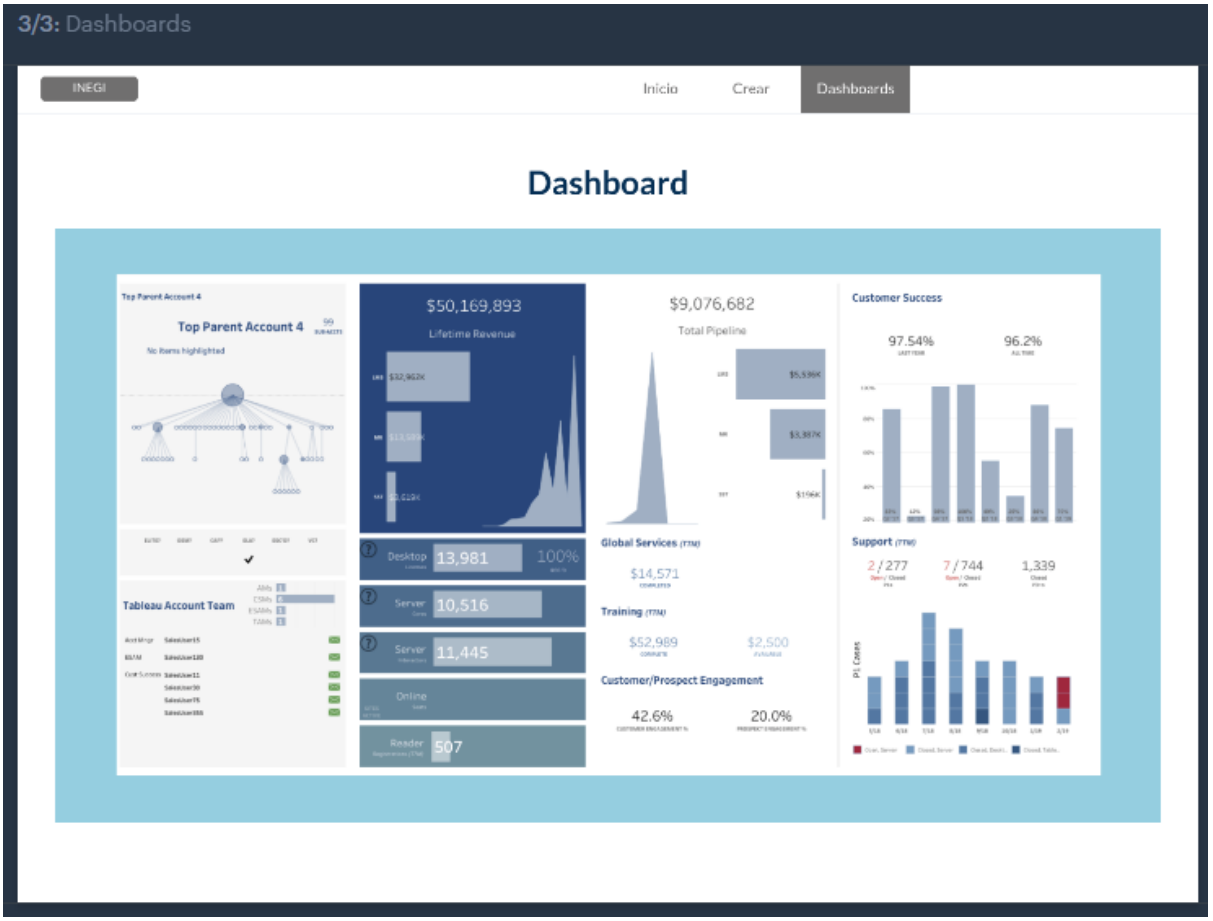
2/3: Crear

INEGI

Inicio | Crear | Dashboards

# Crea tu Propia Gráfica




## Implementación

Más adelante se utilizó el diseño del prototipo para generar la página web utilizando React para poder elaborar la interfaz del usuario, HTML para diseñar los documentos que se muestran en el navegador web, CSS para estilizar los documentos HTML y Javascript para codificar el comportamiento de los elementos de la interfaz de usuario.

# Inicio.js

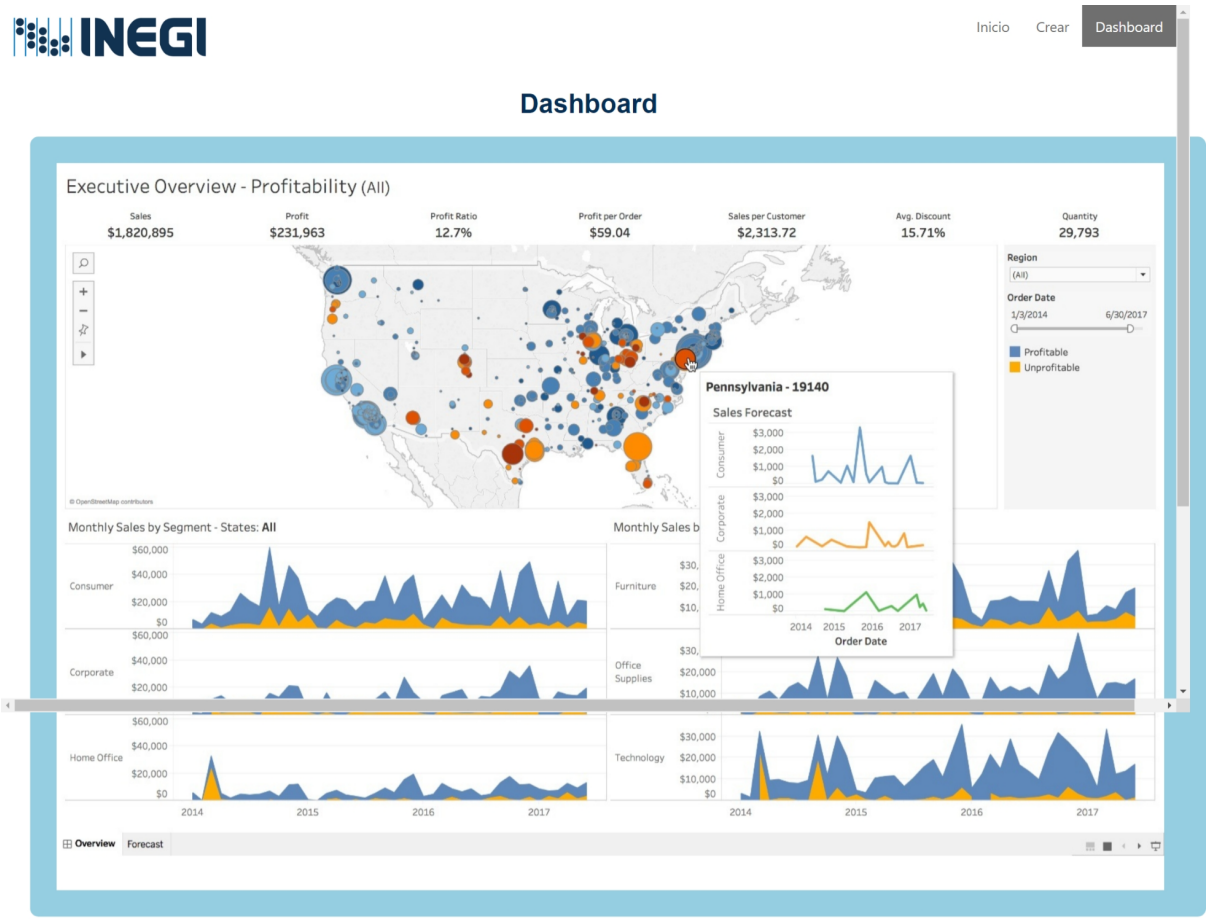
Página que dará una descripción al usuario sobre la funcionalidad de la interfaz y sus objetivos.

# Crear.js

Página que permitirá al usuario crear gráficas con un dataset de entrada y a través de un chatbot en Amazon Lex.

## Dashboard.js

Página que permitirá ver al usuario las gráficas que ha creado.




# Funciones lambda en servidor en la nube

## awslexsdk.py

Esta es una de los scripts que se encuentran en AWS. Este script utiliza el SDK de AWS para crear un bot Lex de forma programática. Para poder crear los tipos de ranuras

específicas para cada dataset, esta función utiliza un arreglo en donde se encuentran las columnas y de esta forma se crea el bot.

```
import boto3 as sdk
import ExperimentalSynonymFeatureForAmazonLex as syn

client=sdk.client('lexv2-models')

class SlotTypeUtils:

   slotID=''
   botID=''
   botVersion=''
   localeID=''

   def __init__(self,clientOJ,slotID,botID,botVersion,localeID):
       '''
       Costructor
           -Parameters:
               clientOJ:The SDK Object
               slotID:The id of the Amazon Lex V2 slot type.
               botID:The id of the Amazon Lex V2 bot.
               botVersion:The version of the Amazon Lex V2 bot(Always DRAFT in developer mode).
               localeID:The id of language used in Amazon Lex V2.
       '''

       self.clientOJ=clientOJ
       self.slotID=slotID
       self.botID=botID
       self.botVersion=botVersion
       self.localeID=localeID
       self.describeSlotType()

   def describeSlotType(self):
       '''
       This function describes a slotType using the atributes in the class,used for updating class.
       '''

       self.slotDescription = self.clientOJ.describe_slot_type(
           slotTypeId=self.slotID,
           botId=self.botID,
           botVersion=self.botVersion,
           localeId=self.localeID
       )




   def updateSlotType(self,arraySlotValues):
       '''
       This function acepts a list of posible values for a slot type

       Return
           -Response: A json formated to a map in python with the response info.

       '''
       valuesSlotType=[]

       for slotValue in arraySlotValues:
           synObj=syn.wordTosynonym(slotValue)
```

```
            if(len(synObj)!=0):

                myValueMap={
                    'sampleValue':{
                        'value':slotValue
                    },
                    'synonyms':syn.wordTosynonym(slotValue)
                }
            else:

                 myValueMap={
                    'sampleValue':{
                        'value':slotValue
                    }
                }


            valuesSlotType.append(myValueMap)

        response=self.clientOJ.update_slot_type(
            slotTypeId=self.slotID,
            slotTypeName=self.slotDescription['slotTypeName'],
            slotTypeValues=valuesSlotType,
            valueSelectionSetting=self.slotDescription['valueSelectionSetting'],
            botId=self.botID,
            botVersion=self.botVersion,
            localeId=self.localeID
        )

        return response

def updateSlotTypeAndBuildBot(clientOJ,slotID,botId_B,botVersion_B,localeID_B,sloteTypeValues):

    '''
    This function is uses to rebuild the bot with the updated changes
    '''

    mySlot=SlotTypeUtils(clientOJ,slotID,botId_B,botVersion_B,localeID_B)
    resp=mySlot.updateSlotType(sloteTypeValues)
    response = clientOJ.build_bot_locale(
    botId=botId_B,
    botVersion=botVersion_B,
    localeId=localeID_B
    )

    return response


def constructBot(array):

    response=updateSlotTypeAndBuildBot(client,'1ARCF9BNKO','3BXJYNJNTT','DRAFT','es_419',array) #Hay
que esperar a que se contruya el bot
    print(response)
```

## ExperimentalSynonymFeatureForAmazonLex.py

Este script utiliza web scrapping para poder generar sinónimos que se van a utilizar en las ranuras de nuestro futuro bot. La página que se utiliza para encontrar estos sinónimos es *https://www.sinonimosonline.com/*.

```
import requests
import bs4 as bs



def wordTosynonym(word):

   try:

       syn = []
       URL=f'https://www.sinonimosonline.com/{word.lower()}'
       page=requests.get(URL)
       miSopita=bs.BeautifulSoup(page.content, "html.parser")
       results=miSopita.find(class_="synonim")
       final_filter=results.find_all("a","sinonimo")

       for synonym in final_filter:
           print("=================")
           print(synonym.text)
           print("=================")
           syn.append(synonym.text)

       formatedList=formatForAmazonLexSDK(syn)

   except:

       #In case word isnt found
       formatedList=[]

   return formatedList

def processStringList(synomyms):

   #For compatibility with other websites
   processStringSyn=synomyms.replace("  ","")
   li=list(processStringSyn.split(","))
   return li

def formatForAmazonLexSDK(synonymsList):

   synonyms=[]

   for synonym in synonymsList:
       synonymMap={
           'value':synonym
       }
       synonyms.append(synonymMap)

   return synonyms
```

## lambda_function.py

Este script es el que se manda a llamar al momento de que se recibe un nuevo dataset en formato .csv en nuestro servidor en nube. Hace un procesamiento del mismo dataset para extraer las ranuras específicas de cada dataset.

```
import json
import pandas as pd
import io
import urllib
import boto3
import requests
import bs4 as bs
from bs4 import BeautifulSoup
import ExperimentalSynonymFeatureForAmazonLex as ex
import time
import urllib.request
from awslexsdk import constructBot

s3 = boto3.client('s3')

def lambda_handler(event, context):

   print(event)


   # Getting bucket
   bucket = event['Records'][0]['s3']['bucket']['name']

   # Getting file name
   name = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])

   print(bucket, name)



   # Fetching file
   try:
       resp = s3.get_object(Bucket=bucket, Key=name)
       df = pd.read_csv(resp['Body'], sep=',')

       columns = []

       for type in df.columns:
           print(type)
           columns.append(type)

       print(columns)
       constructBot(columns)


       return {
           'status': 200,
           'body': columns
       }

   except Exception as e:
       print(e)

       return {
           'status': 400,
           'body': str(e)
       }
```