

# ORT

Consigna vigente unicamente para final del 04/12/2023.-

Alumnos que cursaron con el profesor **Alejandro Villafañe** (no importa el cuatrimestre).-

Este documento se suma a las pautas publicadas en el Aula Virtual.

## Tecnologías que deben / pueden usar

- Git
- Node JS
- Javascript
- Express JS
- Express Validator
- MySQL (para los que vieron base de datos)
- Mocha JS (para los que vieron testing)

---

## Para comenzar a trabajar

Lo primero a realizar es clonar desde Github el proyecto. **Clonar**, **no descargar** el zip desde Github.

```
git clone https://github.com/alezvi/ort-2023-12-04
```

El proyecto contiene el repositorio con el código base para comenzar a trabajar.


Es **muy importante** mantener el proyecto como se entrega y solamente **agregar** las cosas que necesiten, no eliminen ni modifiquen nada de lo que se entrega, conozcan o no lo que hacen.

Una vez descargado van a abrir el archivo `.env.example` que se encuentra en la carpeta raíz y van a completar la parte que se encuentra en blanco y que dice lo siguiente:

NOMBRE=  
APELLIDO=  
EMAIL=  
CUATRIMESTRE=

Cuando lo completen debe quedar algo así:

NOMBRE=Cosme  
APELLIDO=Fulanito  
EMAIL=cosme@fulanito.fox  
CUATRIMESTRE=2023-2

 Este paso **deben completarlo antes** de comenzar, de lo contrario el examen no será tenido como válido. No lo dejen para otro momento.

---

## Al finalizar

**Todos los archivos** deben ser entregados en el zip.

La carpeta **node\_modules** no se sube, si lo hacen resta 1 punto.

Todo lo demás (incluida la carpeta data con todos los .txt) e incluida la carpeta oculta .git se sube. Si falta este punto se desapueba el examen.

Para no equivocarse en esto la recomendación es **borrar la carpeta node\_modules** y luego comprimir todo y subirlo al aula virtual para la entrega. Comprimir en formato **zip** o **rar**.

**IMPORTANTE: no hace push** en ningún momento. Los cambios quedarán solo localmente. Y por seguridad no borres tu copia local hasta que tengas la nota del proyecto.

---

## Recordatorios

El objetivo del examen es poner en ejercicio todos los conocimientos sobre Git, Javascript, y Express, obtenidos durante la cursada. No te confíes de soluciones similares que haya en internet, ni uses herramientas que generen código. Podés buscar ejemplos y ayuda pero no hagas copy paste de ninguna solución ajena. Cuando detectemos estos casos bajaremos puntaje.

Cada tarea va a tener un requerimiento descripto. Cada requerimiento **debe** ser resuelto en una rama de git. Para crear la rama abajo de cada punto requerido indica el nombre que debe llevar. Al finalizar la consigna se deben realizar los commits correspondientes.

Recuerden que **la entrega oficial es a traves de aula virtual**, el **Github no es valido** ante la institución, pero queda como una opción para ayuda a mantener el trabajo y evitar que se pierda, ademas de ayudar a los profesores a poder corregir mejor las consignas resueltas.

Se solicita usar **commits atómicos**, o sea 1 commit por archivo, describiendo lo realizado. Si no se respeta esta forma en todos los commits, se descontarán 2 (dos) puntos de la calificación final.

El diseño no es relevante. Solo la funcionalidad es necesaria.

En esta consigna hay 4 puntos a resolver. Como mínimo 2 (dos) tienen que estar completos y correctamente resueltos. Los 2 (dos) restantes son opcionales, pero suman en caso de que puedas resolverlos.

---

## Trabajo a realizar

Vamos a desarrollar una interfaz que se encarga de analizar textos escritos.

En la carpeta raiz del proyecto hay un archivo llamado "lorem-ipsum.txt" donde pueden encontrar un texto de ejemplo para hacer pruebas.

---

### 1 - Ingresar un texto

URL: `/texts`

Método: `POST`

Cuando ingreso a la URL `/texts` por el método `GET` el navegador mostrará un formulario de HTML con un campo de entrada texto (textarea). Al enviarlo a la misma URL `/texts` pero por el método `POST` el texto será recibido y guardado localmente en un archivo.

Como se guarda en un archivo ? No tienes que hacer nada, solamete usar la funcionalidad provista en `/utils/utils.js` Llamando a la función de la siguiente manera conseguirás el id y se guardará en la carpeta `data` del proyecto. El id es el nombre del archivo sin la extensión `.txt`

```
let id = utils.save('lorem ipusm')
```

Aquí es donde el usuario puede ingresar un texto para que sea guardado en el sistema y a continuación pedir información analítica al respecto.

### Requerimientos

- El texto debe contener un mínimo de 20 palabras
- Entre las 10 palabras como mínimo debe haber 100 caracteres
- El texto no puede superar los 10000 caracteres

### Respuesta

Devuelve un objeto conteniendo el `id` que es el identificador con el que mas tarde podremos pedir información analítica sobre el texto entregado en el post.

```
{
  "id" : "1234567890"
}
```

Nombre para la rama: `git commit -m "ingresar-texto"`

---

## 2 - Contar caracteres

URL: `/texts/{id}`

Metodo: `GET`

Como busco un texto por el id ? No tienes que hacer nada, solamente usar la funcionalidad provista en `/utils/utils.js` Llamando a la función de la siguiente manera conseguirás el texto y podras hacer el analisis correspondiente para devolver la respuesta solicitada abajo. El id es el nombre del archivo sin la extensión `.txt`

```
let text = utils.read('367e2e07-cb60-42b7-a3b8-0734af939a28')
```

Invocando al endpoint `texts` con el `id` del texto despues de la barra vamos a recibir como respuesta un mensaje JSON como el siguiente ejemplo. (Todos los valores de la derecha son de ejemplo)

```
{
  "total_length" : 418,
  "words_count" : 76,
  "paragraphs_count": 3,
}
```

- `total_length` es la cantidad de caracteres del texto
- `words_count` es la cantidad de palabras del texto
- `paragraphs_count` es la cantidad de párrafos del texto

✓ TIP: Las palabras se separan por espacio. Los párrafos se separan por puntos.

Nombre para la rama: `git commit -m "contar-caracteres"`

### 3 - Párrafos que contienen palabras

Invocando al endpoint podemos obtener una lista de los párrafos que contienen una determinada palabra. Por ejemplo, si busco la palabra "arbol" voy a invocar la url `phrases?q=arbol` pasando en el query string de la URL el termino concreto de busqueda.

URL: `/phrases`

Metodo: `GET`

Ejemplo: `/words?q=arbol`

#### Respuesta

```
{
  "query" : "arbol",
  "matches_count" : 2,
  "matches" : [
    "...frente a la casa habia un arbol...",
    "...las hojas del arbol aun estaban verdes..."
  ]
}
```

Es un objeto que contiene tres claves. La primera clave es `query` y repite el valor del parametro `q` ingresado por el usuario en la URL. El segundo es `matches_count` que contiene la cantidad de coincidencias obtenidas. El tercero es `matches` con la lista (array) de parrafos coincidentes donde se encuentra la palabra buscada, "arbol" en este ejemplo.

Nombre para la rama: `git commit -m "busqueda-palabras"`

---

## 4 - Devolver una frase

Invocando por GET la URL `/phrases/{id}/random` responderá con un string que sea una frase cualquiera del texto ingresado. En `{id}` se reemplaza por el id real de una frase ingresada previamente.

### Respuesta:

```
{
  "id" : 1234567890,
  "phrase" : "...esta es una frase elegida al azar y por lo tanto en cada solicitud se leera
otra frase o tal vez la misma... asi funciona el azar..."
}
```

Nombre para la rama: `git commit -m "frase-random"`

---

Consultas ? Escribir a [alejandro.villafane@ort.edu.ar](mailto:alejandro.villafane@ort.edu.ar)