UNIVERSIDAD TÉCNICA NACIONAL

INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN

SEDE DEL PACÍFICO

GUÍA DE ACTIVIDAD PRÁCTICA Y/O LABORATORIO

Curso:	ITI-621 – Tecnologías y Sistemas Web III			**Profesor:**	Jorge Ruiz (york)
Puntos por Ganar:	75			**Fecha:**	20/Noviembre/2023
Puntos Obtenidos:	0	**Porcentaje:**	15%	**Nota**:	0

- **Objetivos de la actividad.**

  - Evaluar la ejecución de conocimiento en forma individual del alumno referente a la implementación de soluciones aplicando la infraestructura de Google Firebase:
    - Capacidad de desarrollar deploys.
    - Autenticación de usuarios.
    - Carga de datos a la Firestore Database.
    - Consulta sobre los datos almacenados y generar las salidas requeridas.

- **Instrucciones de la prueba.**

Segunda Prueba Parcial:

Observaciones:

- Usted tiene derecho a consultar a su profesor sobre cualquier duda con respecto al planteamiento del problema, o de lo que se espera del producto final, dudas relacionadas, con el código escrito, o con el almacenamiento de la prueba, no serán sujetas de atención.

- Por la naturaleza de la prueba, o sea práctica, la misma cuenta con el beneficio de utilizar material de apoyo, en otras palabras, código visto en clases, libros, internet. Pero el material es de uso exclusivo de cada estudiante, no es transferible a ninguno de sus compañeros.

- La prueba es de carácter individual, por favor apéguese a esta solicitud.

- El tiempo es valioso, y lamentablemente corto, por lo que se le sugiere, que cualquier distractor, como escuchar música, atender el celular o salidas del aula por la razón que fuese, trate de limitarlas solo a emergencias, **no está permitido el uso de WhatsApp**.

- Al finalizar la prueba el estudiante deberá de entregar un documento .zip con el contenido del código fuente, así como un archivo .txt con la ruta del sitio publicado para validar el funcionamiento de este, agregue los datos de un usuario para facilitar la consulta.

Apoyo al estudiante

1. En el proyecto que usted ya tiene corriendo contra **Google Firebase**, del cual usted utiliza varios recursos, entre ellos **Authentication, FireStore, Store y Hosting**, agregue en su carpeta **public** una página html llamada **loadData.html,** recuerde aplicar el formato que ya se tiene establecido para las páginas de este proyecto, y una vez creada la página agregue las siguientes líneas de código.

```
<!-- update the version number as needed -->
<script defer src="/__/firebase/10.5.2/firebase-app-compat.js"></script>
<script defer src="/__/firebase/10.5.2/firebase-auth-compat.js"></script>
<script defer src="/__/firebase/10.5.2/firebase-database-compat.js"></script>
<script defer src="/__/firebase/10.5.2/firebase-firestore-compat.js"></script>

<!-- initilized firebase object -->
<script defer src="/__/firebase/init.js"></script>
```

```
<h3>Carga de datos</h3>
<div>
    <label for="txtCSV">Filename:</label>
    <input type="file" name="txtCSV" id="txtCSV">
    <br> <br>

    <input type="button" name="btnLoad" id="btnLoad" value="Subir archivo" />
</div>
```

```
<!-- load js code to storage data  -->
<script defer src="js/support/bc_csvToCollection.js"></script>
```

Correspondientemente a como lo hemos realizado en clases.

2. Además, debe agregar en su carpeta JS (javaScript) una carpeta llamada **support** y dentro de esta, debe crear un archivo **bc_csvToCollection.js** y el cual contendrá el siguiente código:

```
// JavaScript Document
var db = firebase.apps[0].firestore();

const txtCSV = document.querySelector('#txtCSV');
const btnLoad  = document.querySelector('#btnLoad');

btnLoad.addEventListener('click', function(){
    lecturaCSV(txtCSV.files[0]).then(r => {
        txtCSV.value = '';
    });
});

async function lecturaCSV(archivo){
    const nomarch = archivo.name.split('.')[0];
    const lector = new FileReader();
    lector.readAsText(archivo);
    await esperarSegundos();
```

```
    if(lector.result != null){
        let data = lector.result.split('\n');
        let etiquetas = data[0].split(';');

        for (let index = 1; index < data.length; index++) {
            const valores = data[index].split(';');
            let salida =  {}
            for (let index2 = 0; index2 < etiquetas.length; index2++) {
                salida[etiquetas[index2]] = valores[index2];
            }

            db.collection(nomarch).add(salida).then(function(docRef) {
                console.log("ID del registro: " + docRef.id);
            }).catch(function(FirebaseError) {
                console.log("Error al registrar el dato: " + FirebaseError);
            });

        }
    }
}

function esperarSegundos() {
    return new Promise((resolve) => {
        setTimeout(() => {
            resolve('resolved');
        }, 5000);
    });
}
```

3. Descargará el archivo .zip adjunto a este proyecto, que contiene los datos de la base de datos de Nortwhind, en formato csv para cada tabla requerida.

¿Qué se requiere en la prueba?

1. Modificará el menú principal para poder llamar a la página de **loadData.html**, recuerde además que esta página solo puede mostrarse si el usuario esta autenticado.

2. Probar que la página es capaz de cargar los datos dentro del **FireStore**, por lo que es necesario crear, las reglas necesarias para hacer la carga adecuadamente.

3. El código dado por su profesor **bc_csvToCollection.js** para la carga de datos, tiene el inconveniente de que todos los datos recuperados, los trata como texto y existen datos numéricos como enteros y flotantes que no son procesados adecuadamente, por lo que debe aplicar las correcciones necesarias para que dichos datos se procesen como tal su intención.

Se agrega el siguiente enlace para que observe algunas funciones de tipo **Number** que le pueden ayudar a corregir este problema:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number

Esta situación mencionada anteriormente pude afectar también los datos de fecha, por lo que también se le brinda el siguiente enlace para que ejecute las correcciones pertinentes con el sudo del objeto **Date**:

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Date

Propuesta debe desarrollar entonces una función que reciba el dato en String, pero que retorne el dato en sí en su formato nativo [Entero | Flotante | Fecha | Cadena].

En caso de que exista algún tipo de valor faltante, o que venga en blanco o con el valor NULL, convierta la salida a la cadena "na".

4. Creará una página llamada **consuFactura.html**, igual debe seguir las restricciones de verse idéntica en formato al resto de las páginas del sitio, ser invocada desde el menú principal y solo será visualizada si el usuario se encuentra registrado ante el sitio.

   Esta página debe ser capaz de consultar el número de una factura y presentar los datos relacionadas a esta de la siguiente forma:

Factura #: 999999

Cliente:	El nombre del cliente.
Contacto:	El titulo y nombre completo del contacto.
Destino:	El país, ciudad y código postal.

Facturada:	dd/MMM/yyyy
Requerida:	dd/MMM/yyyy
Despachada:	dd/MMM/yyyy
Empleado:	Nombre Completo

Código	Nombre	Cantidad	Precio Uni	Descuento	Total
				Total	**Suma totales**

- **Evaluación de la prueba.**

Ítem	Descripción	Puntos
1	Invocar la página **loadData.html** desde el menú principal del sitio ya creado.	05
2	Validar que la página loadData.html solo cargue si el usuario ya se encuentra autenticado.	05
3	Crear las reglas del FireStore para: Customers Employees Products Orders OrdersDetails	 05 05 05 05 05
4	Función de retorno de valor String a nativo: A entero A Flotante A Fecha Control de faltante de dato	 05 05 05 05
5	Generalidades de la página consuFactura.html	05
6	Consultar la factura, permitiendo una salida como la presentada en el ejemplo	15
	Puntos por ganar	**75**