



MANUAL DEL DESARROLLADOR

PROYECTO COCINAPP

Manual del desarrollador

Índice de contenido

Estructura de carpetas del código fuente:	2
Documentación del código:.....	3
Instrucciones para hostear la página en un servidor:	5
Cambio de estado editable de los pedidos (Python)	6
Credenciales de firebase (nueva base de datos).....	7
Cuentas de correo para el login:	9
Como crear una nueva cuenta en Authentication:	10
Envío de correo al usuario al cambiar estado del pedido:	11
Correo correspondiente en contacto.html.....	12

Estructura de carpetas del código fuente:



css	19/04/2024 6:30
docs	23/04/2024 3:13
images	19/04/2024 6:30
js	23/04/2024 2:30
pages	19/04/2024 6:30
index.html	17/04/2024 18:39
jsdoc.conf.json	

css: contiene el archivo style.css.

docs: contiene todos los archivos jsdoc generados.

images: contiene todas las imágenes locales utilizadas por el código (fondos, iconos...).

js: contiene todos los archivos .js del código (archivos de funciones, de eventos, de configuracion...).

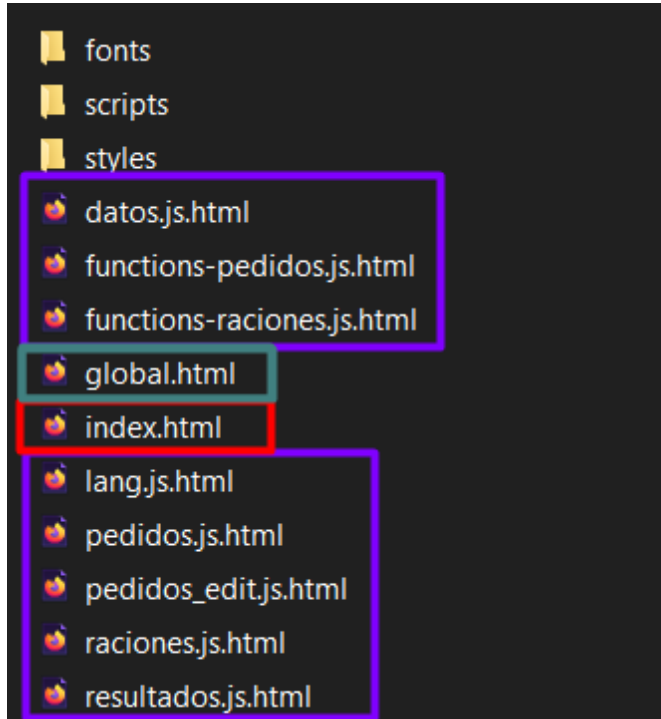
pages: contiene todos los archivos .html a excepción de index.html.

index.html: archivo .html para la página principal de nuestra pagina web.

jsdoc.conf.json: configura la generación de código por jsdoc. Aquí hemos configurado que cree la documentación para los archivos de la carpeta js y añada a documentación en docs.

Documentación del código:

Hemos utilizado jsdoc para comentar las funciones de nuestro código. En la carpeta docs se encuentran todos los archivos generados por jsdoc:



Archivos de código fuente y sus funciones: en estos archivos se muestra el código fuente de cada archivo .js, con la lista de funciones que lo componen a la derecha:

Source: functions-pedidos.js

LISTA DE FUNCIONES EN FUNCTIONS-PEDIDOS.JS

Home

Global

```

1. import { db, ref, get, set, child, update, remove, onValue } from "./firebase-config.js";
2. /**
3.  * Selecciona todos los datos de todos los pedidos y llama a una función de devolución de llamada con los datos a
4.  * @param {function} callback La función de devolución de llamada que se llama con los datos actualizados.
5.  * @throws {Error} Lanza un error si hay algún problema al recuperar los datos de la base de datos.
6.  */
7.
8. function selectAllDataPedidos(callback) {
9.   const dbref = ref(db);
10.
11.   onValue(child(dbref, "pedidos/"), (snapshot) => {
12.     const data = [];
13.
14.     snapshot.forEach((childSnapshot) => {
15.       const key = childSnapshot.key;
16.       const rowData = childSnapshot.val();
17.
18.       // Verifica si el pedido tiene editable false
19.       if (rowData.editable === false) {
20.         // Check if rowData.detalles is defined before mapping over it
21.         const detalles = rowData.detalles ? rowData.detalles.map((detalle) => ({
22.           racion: detalle.racion,
23.           cantidad: detalle.cantidad,
24.           precio: detalle.precio,
25.         })) : [];
26.         const pedidoData = {

```

actualizarDatosFirebase
 buscarEnAlergenos
 buscarEnDetalles
 comprobarMaximo
 deleteDataPedidos
 deleteDataRaciones
 esCero
 esPositivo
 insertDataRaciones
 introducirDatosFirebase
 isValidInteger
 noCoincide
 parseAlergenos
 sanitizeInput
 searchDataPedidos
 searchDataRaciones
 selectAllDataPedidos
 selectAllDataRaciones
 selectAllDataResults
 selectData
 selectDataUsuario
 sendEmailRecoger
 sendEmailRecogido
 updateData
 updateDataPedidos

global.html:

Se puede encontrar la lista de todas las funciones en index.html o global.html:

Global

LISTA DE FUNCIONES EN TODO EL CODIGO

Home

Global

actualizarDatosFirebase
 buscarEnAlergenos
 buscarEnDetalles
 comprobarMaximo
 deleteDataPedidos
 deleteDataRaciones
 esCero
 esPositivo
 insertDataRaciones
 introducirDatosFirebase
 isValidInteger
 noCoincide
 parseAlergenos
 sanitizeInput
 searchDataPedidos
 searchDataRaciones
 selectAllDataPedidos
 selectAllDataRaciones
 selectAllDataResults
 selectData
 selectDataUsuario
 sendEmailRecoger
 sendEmailRecogido
 updateData
 updateDataPedidos

Methods

(async) actualizarDatosFirebase(selectedFile, id, idOriginal, variableDescripcion, variableAlergenos, variablePrecio, variableStock, variableMaximo)

Actualiza los datos de una ración en Firebase, incluida la imagen asociada, si se proporciona una nueva imagen.

Parameters:

Name	Type	Description
selectedFile	File	El archivo de imagen seleccionado para la ración (puede ser undefined si no se cambia la imagen).
id	HTMLInputElement	El campo de entrada HTML que contiene el ID de la ración actualizada.
idOriginal	HTMLInputElement	El campo de entrada HTML que contiene el ID original de la ración.
variableDescripcion	HTMLInputElement	El campo de entrada HTML que contiene la descripción actualizada de la ración.
variableAlergenos	HTMLInputElement	El campo de entrada HTML que contiene los alérgenos actualizados de la ración.
variablePrecio	HTMLInputElement	El campo de entrada HTML que contiene el precio actualizado de la ración.
variableStock	HTMLInputElement	El campo de entrada HTML que contiene el stock actualizado de la ración.
variableMaximo	HTMLInputElement	El campo de entrada HTML que contiene el máximo actualizado de la ración.

Source: [functions-raciones.js, line 350](#)

buscarEnAlergenos(alergenos, query) → {boolean}

index.html: pagina home de la documentación jsdocs. Muestra una introducción al proyecto y enlaza con las funciones de global.html:

Home

ENLACES A LAS FUNCIONES DE GLOBAL

Home

Global

actualizarDatosFirebase
 buscarEnAlergenos
 buscarEnDetalles
 comprobarMaximo
 deleteDataPedidos
 deleteDataRaciones
 esCero
 esPositivo
 insertDataRaciones
 introducirDatosFirebase
 isValidInteger
 noCoincide
 parseAlergenos
 sanitizeInput
 searchDataPedidos
 searchDataRaciones
 selectAllDataPedidos
 selectAllDataRaciones
 selectAllDataResults
 selectData
 selectDataUsuario
 sendEmailRecoger
 sendEmailRecogido
 updateData
 updateDataPedidos

INTRODUCCION DEL HOME

cocinApp

Bienvenido a **cocinApp**, una plataforma web diseñada para agilizar la gestión de pedidos y raciones en tu cocina.

Descripción del Proyecto

cocinApp es una página web utilizada por la cocina del CIP Estella para la creación, edición y visualización de raciones que pueden ser solicitadas por los clientes a través de una aplicación móvil dedicada. Además de la gestión de raciones, la plataforma permite:

- Gestión de Pedidos:** Visualización y edición de los pedidos realizados a través de la aplicación móvil.
- Análisis de Datos:** Generación de gráficos que muestran información sobre los pedidos realizados a lo largo de semanas, meses y años. Esto proporciona una visión general del flujo de pedidos y patrones de consumo a lo largo del tiempo, lo que ayuda a tomar decisiones informadas sobre inventario y estrategias de oferta.

Características Principales

- Creación y edición de raciones.
- Gestión de pedidos realizados desde la aplicación móvil.
- Análisis y visualización de datos sobre los pedidos a lo largo del tiempo.

Manual del usuario

[Link al manual de usuario.](#)

Manual del desarrollador

Instrucciones para hostear la página en un servidor:

La documentación detalla sobre la instalación de la página en el servidor se encuentra en el archivo:

[Manual_de_desarrollador_servidor.pdf](#)

Cambio de estado editable de los pedidos (Python)

Un pedido creado por un cliente pasara una hora en estado editable para que el cliente pueda editar o eliminar el pedido. Pasado este tiempo, solo cocina podrá editar ciertos datos como el estado, la fecha y la hora, o borrar el pedido.

Para controlar esto, se utiliza una tarea en Python que comprobara cada hora con la base de datos qué pedidos se deben cambiar a no editables y cambiara el valor editable a false.

Solo los pedidos con editable = false aparecerán en la lista de pedidos de nuestra página.

La documentación detalla sobre el funcionamiento e instalación de dicha tarea se encuentra en el archivo:

[AppCocina_Tarea_actualizar_edicion_Pedido.pdf](#)

Credenciales de firebase (nueva base de datos)

Si en algún momento se cambia la base de datos de firebase que la aplicación esta utilizando, debemos darle las credenciales de conexión para esa base de datos especifica en el archivo `firebase-config.js`:

```

15 firebase-config.js > ...
//FUNCIONES IMPORTADAS
import { initializeApp } from "https://www.gstatic.com/firebasejs/10.7.0/firebase-app.js";
import { getAnalytics } from "https://www.gstatic.com/firebasejs/10.7.0/firebase-analytics.js";
import { getStorage } from "https://www.gstatic.com/firebasejs/10.7.0/firebase-storage.js";
import { getDatabase, ref, get, set, child, update, remove, onValue, } from "https://www.gstatic.com/firebasejs/10.7.0/firebase-database.js";

// CONFIGURACION DE LA BASE DE DATOS DE FIREBASE
const firebaseConfig = {
  apiKey: " ",
  authDomain: " ",
  databaseURL: "https:// ",
  projectId: " ",
  storageBucket: " ",
  messagingSenderId: " ",
  appId: " ",
  measurementId: " "
};

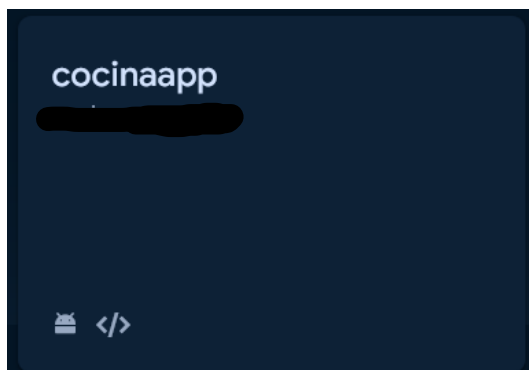
//INICIARANDO FIREBASE
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
const db = getDatabase();

//OBTENER REFERENCIA AL ALMACENAMIENTO DE FIREBASE
const storage = getStorage(app);//esto es lo nuevo

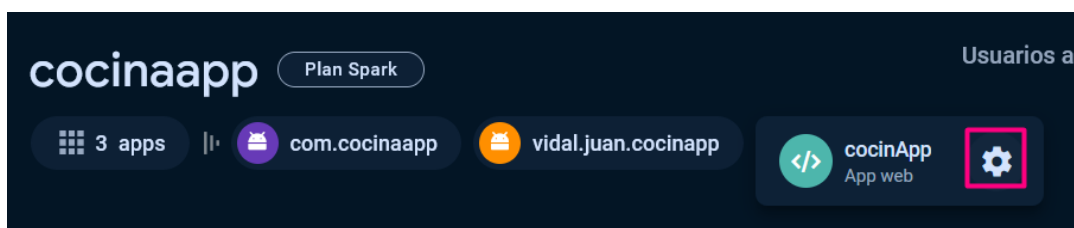
// EXPORTAMOS LOS OBJETOS DE FIREBASE PARA USARLOS EN OTROS FICHEROS
export { app, analytics, storage, db, ref, get, set, child, update, remove, onValue };

```

Para encontrar estas credenciales en firebase, se selecciona el proyecto:



Seleccionar la app web e ir a opciones:



Bajamos, y copiamos la constante firebaseConfig:

Luego, inicializa Firebase y comienza a usar los SDK de los productos que quieres utilizar.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  authDomain: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.com",
  databaseURL: "https://XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.firebaseio.com",
  projectId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  storageBucket: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  messagingSenderId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  appId: "1:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  measurementId: "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Cuentas de correo para el login:

Para iniciar sesión en CocinApp, se utilizan los correos especificados en el archivo auth.js:

```
//Aquí poner usuarios Administradores que esten registrados en la base de datos auth de firebase.  
const admin_email = ["[redacted]@gmail.com", "[redacted]@gmail.com"];
```

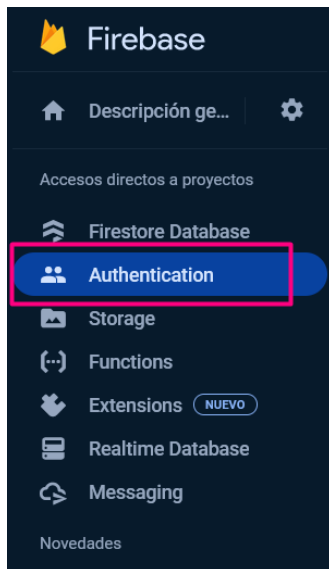
Como dice el código, estos correos son cuentas de usuarios administradores registrados en la base de datos Authentication en el firebase. La contraseña será la contraseña del usuario en Authentication en firebase.

Si se pierde la contraseña de estas cuentas, desde la cuenta de firebase se pueden crear usuarios nuevos en Authentication y borrar los antiguos.

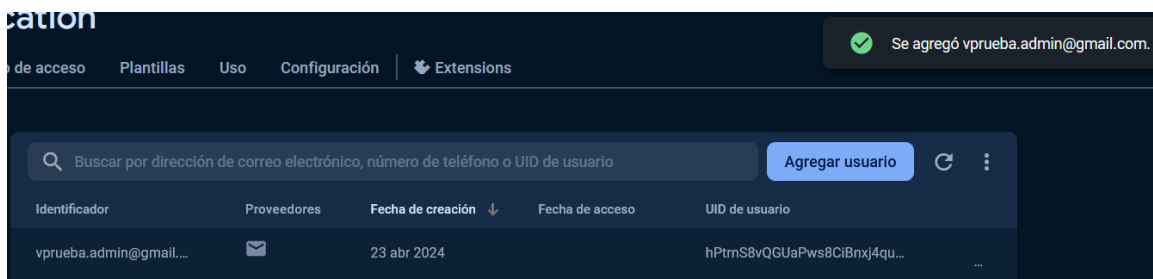
En caso de crear nuevos usuarios habría que cambiar los usuarios de “admin_email” en nuestro código por los nuevos. El código esta creado para manejar cuentas de usuario en forma de correos electrónicos, por lo que los nuevos usuarios deben tener el mismo formato.

Como crear una nueva cuenta en Authentication:

En la cuenta de firebase vamos a Authentication:



En Authentication, pulsamos en “Agregar usuario”, añadimos el correo que queremos usar y la contraseña, y le damos a agregar:



Ahora se debe añadir ese correo a auth.js, en admin_email, y ya podremos usar nuestro nuevo usuario para iniciar sesión en el login.

Envío de correo al usuario al cambiar estado del pedido:

Cuando en pedidos.html cambiamos el estado de un pedido a recoger o recogido, se utiliza mailto para abrir la aplicación de correo predeterminada de nuestro equipo y generar un correo con el cliente como destinatario, notificándole de los cambios en su pedido.

```
function sendEmailRecogido(senderEmail, recipientEmail, idPedido) {  
  console.log("enviando correo");  
  
  // Asunto y cuerpo del email  
  const subject = 'Su ha sido recogido';  
  const body = `Su pedido con el codigo ${idPedido} ha sido recogido.`;  
  
  // Creando el mailto URI  
  const mailtoURI = `mailto:${recipientEmail}?subject=${encodeURIComponent(subject)}&body=${encodeURIComponent(body)}&from=${senderEmail}`;  
  
  // Abre el cliente email por defecto con el email pre-hecho  
  window.open(mailtoURI, '_self');  
}
```

Es importante que en nuestra aplicación de correo predeterminada estemos logueados con el correo de cocina para que al generar este email utilice el correo de cocina como remitente.

A pesar de que en el código le damos como remitente la cuenta de cocina desde pedidos.js, varios servicios de correo solo permiten enviar correos del remitente que haya iniciado sesión en el servicio, por lo que es importante tenerlo en cuenta.

Correo correspondiente en contacto.html

Hay que tener en cuenta cambiar el correo que aparece en contacto.html si queremos que la cuenta de contacto sea otra:



```
<!-- Botón de email con icono -->  
<a href="mailto:[redacted]@educacion.navarra.es" class="btn btn-outline-secondary mb-2">  
  <i class="bi bi-envelope"></i>  
</a>  
<input type="email" class="form-control" id="email" value="[redacted]@educacion.navarra.es" readonly>  
</div>
```