



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones WEB
Informática y Comunicaciones

Gestor de Dietas

Tutor individual: José Luis Román Bienes

Tutor colectivo: José Luis Román Bienes

Año: 2023

Fecha de presentación: 27/06/2023

Nombre y Apellidos: Alberto Velázquez Rapado

Email: alberto.velrap@educa.jcyl.es



Tabla de contenido

1	Identificación proyecto	4
2	Descripción general del proyecto	4
2.1	Objetivos	4
2.2	Entorno de trabajo (tecnologías de desarrollo y herramientas)	4
3	Descripción general del producto.....	5
3.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.	5
3.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.	5
3.2.1	Base de Datos	5
3.2.2	BackEnd.....	6
3.2.3	FrontEnd	11
3.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.....	13
4	Planificación	14
5	Documentación Técnica: análisis, diseño, implementación y pruebas.	14
5.1	Especificación de requisitos	14
5.2	Análisis del sistema	15
5.3	Diseño del sistema:	15
5.3.1	Diseño de la Base de Datos.....	15
5.3.2	Diseño de la Interfaz de usuario.	16
5.3.3	Diseño de la Aplicación.	23
5.4	Implementación:.....	23
5.4.1	Entorno de desarrollo.	23
5.5	Pruebas.	23
6	Manuales de usuario	23



6.1	Manual de usuario	23
6.1.1	Acceso	23
6.1.2	Registro	24
6.1.3	Clientes	26
6.1.4	Dietas	31
6.1.5	Ingestas	34
6.1.6	Configuración.....	41
6.2	Manual de instalación.....	45
7	Conclusiones y posibles ampliaciones	45
8	Bibliografía	45

1 Identificación proyecto

Este proyecto busca aplicar todos los conocimientos que he ido adquiriendo a lo largo de los últimos dos años a la hora de diseñar, programar y desplegar una página web con todo lo que ello conlleva.

En este caso, la aplicación que buscamos desarrollar consiste en un gestor de dietas enfocado al uso por parte de profesionales. Para ello, esta aplicación permitiría que los profesionales puedan crear dietas para distintos clientes. Es decir, por un lado, tendríamos a los profesionales, por otro tendríamos a los clientes de los profesionales, y por otro lado toda la base de datos con los datos nutricionales de los alimentos para poder crear las dietas.

2 Descripción general del proyecto

2.1 Objetivos

Para el desarrollo de este proyecto nos hemos planteado los siguientes objetivos.

A nivel profesional:

- Ser capaz de desarrollar una aplicación completa desde el comienzo has el final del proyecto.
- Conocer y utilizar nuevas tecnologías que desconozco al comienzo del proyecto.
- Aprender a separar el BackEnd del FrontEnd a través del uso de una API REST.
- Profundizar en el proceso de despliegue de una página web.
- Comprender y aplicar el proceso de “empaquetado” de la aplicación.

A nivel personal:

- Aprender a tratar con los problemas que conlleva trabajar con tecnologías desconocidas.
- Gestionar el tiempo necesario para desarrollar una aplicación completa.
- Crear una aplicación funcional.

2.2 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Los medios de los que disponemos para realizar el proyecto son los siguientes:

Software:

- IDE: Visual Studio Code
- Prototipado: Figma
- Base de Datos: MySQL
- BackEnd: NodeJS, ExpressJS, REST API, JSON Web Token

- FrontEnd: Vue, Vuelidate, Vue-Router, Vite, Axios, Pinia, ChartJS, Bootstrap, SASS
- Empaquetado: Docker y NPM
- Despliegue: Docker y Nginx
- Repositorio: Git

3 Descripción general del producto

3.1 Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.

Para el funcionamiento requerido, se precisa de la presencia tanto de un servidor como de un cliente, incluso en el caso en que ambos se encuentren alojados en el mismo equipo.

Los límites del sistema se establecen de la siguiente manera: MySQL y NodeJS tienen la capacidad de gestionar aproximadamente 10^4 peticiones por segundo. Por otro lado, tal y como hemos configurado el frontend tenemos a Nginx entregando la distribución de Vue, y este puede llegar a manejar alrededor de 10^5 peticiones por segundo.

Las funcionalidades básicas de la aplicación son las siguientes:

- Registro de nuevos usuarios
- Acceso a la aplicación
- Modificar y Eliminar al usuario.
- Crear, modificar, eliminar y leer datos de los clientes del usuario.
- Crear, modificar, eliminar y leer datos de las dietas del cliente.
- Crear y leer pesos del cliente.

La aplicación del lado servidor es compatible con cualquier sistema operativo que sea compatible con Docker, lo que incluye Windows, MacOS y diversas distribuciones de Linux.

Por otro lado, la aplicación del lado cliente puede ejecutarse en cualquier navegador que admita JavaScript ES6, HTML5 y CSS.

3.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

3.2.1 Base de Datos

En la Base de Datos al crear las tablas, nos hemos asegurado que las foreign keys hagan un borrado en cascada cuando sean eliminadas.

```

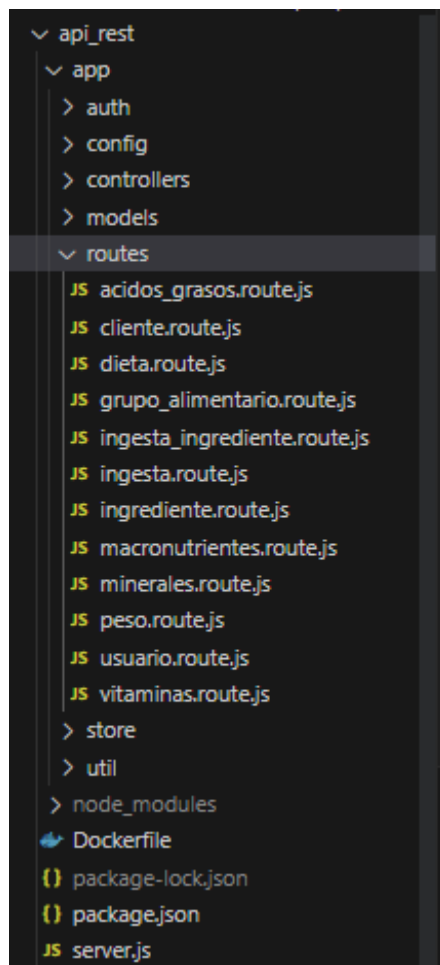
CREATE TABLE IF NOT EXISTS `dietariodb`.`cliente` (
  `id_cliente` INT NOT NULL AUTO_INCREMENT,
  `id_user` INT NULL DEFAULT NULL,
  `nombre` VARCHAR(64) NOT NULL,
  `apellidos` VARCHAR(64) NOT NULL,
  `fecha_nacimiento` DATE NOT NULL DEFAULT (CURRENT_DATE),
  `imagen` VARCHAR(255) NULL DEFAULT 'https://c8.alamy.com/zooms/9/80d94c5b96c54446b2dc609a62b9f61b/2c5xkmf.jpg',
  PRIMARY KEY (`id_cliente`),
  INDEX `id_user` (`id_user` ASC) VISIBLE,
  CONSTRAINT `cliente_ibfk_1`
    FOREIGN KEY (`id_user`)
      REFERENCES `dietariodb`.`usuario` (`id_user`)
    ON DELETE CASCADE
)
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```

3.2.2 BackEnd

A la hora de crear el BackEnd hemos buscado seguir la arquitectura de Modelo Vista Controlador (MVC). Estando los modelos la carpeta Models, los controladores en la carpeta Controllers y las vistas en la carpeta Routes.

models	controllers
JS acidos_grasos.model.js	JS acidos_grasos.controller.js
JS cliente.model.js	JS cliente.controller.js
JS db.js	JS dieta.controller.js
JS dieta.model.js	JS grupo_alimentario.controller.js
JS grupo_alimentario.model.js	JS ingesta_ingredientes.controller.js
JS ingesta_ingredientes.model.js	JS ingesta.controller.js
JS ingesta.model.js	JS ingrediente.controller.js
JS ingrediente.model.js	JS macronutrientes.controller.js
JS macronutrientes.model.js	JS minerales.controller.js
JS minerales.model.js	JS peso.controller.js
JS peso.model.js	JS usuario.controller.js
JS usuario.model.js	JS vitaminas.controller.js
JS vitaminas.model.js	
> routes	> models



También hemos creado unos ficheros de configuración para crear la conexión a la base de datos.

```
const mysql = require("mysql2");
const dbConfig = require("../config/db.config.js");

const conn = mysql.createConnection({
  host: dbConfig.HOST,
  user: dbConfig.USER,
  password: dbConfig.PASSWORD,
  database: dbConfig.DB,
  port: dbConfig.PORT,
  multipleStatements: true,
});

conn.connect(function (err) {
  if (!err) console.log("Connected!");
  else console.log("Connection failed: " + err);
});

module.exports = conn;
```

Y para la utenticación hemos creado una función con JSON Web Token que se encarga de verificar un token con la información del usuario.

```
const jwt = require("jsonwebtoken");
const Usuario = require("../models/usuario.model");

// Función de verificación de ID
const isAuthenticated = (req, res, next) => {
  // Obtenemos el token
  const token = req.headers.authorization;
  // Si no hay token devolvemos error 403 (Forbidden)
  if (!token) {
    return res.sendStatus(403);
  }

  // Verificamos el token, necesitamos el secreto con
  // el que lo codificamos
  jwt.verify(token, process.env.JWT_SECRET || "mi-secreto", (err, decoded) => {
    // Buscamos en Usuario el ID decodificado
    Usuario.findById(decoded._id, (err, data) => {
      if (err) {
        return res.sendStatus(403);
      }
      if (data) {
        // guardamos el ID del usuario en la request
        req.user = data;
        next();
      }
    });
  });
};

module.exports = {
  isAuthenticated,
};
```

Para generar este token hemos creado una función que crea y encripta un token.

```
// Funcion del Token
const signToken = (_id) => {
  return jwt.sign({ _id }, process.env.JWT_SECRET || "mi-secreto", {
    expiresIn: 60 * 60 * 24 * 365,
  });
};
```

Aquí podemos ver como hemos securizado las rutas (Vistas) con la función que hemos nombrado previamente. También podemos ver que cada ruta llama a una función creada en el controlador.


```
module.exports = (app) => {  
  const { isAuthenticated } = require("../auth/index");  
  const { VERSION, REST, API } = require("../util/constants");  
  const cliente = require("../controllers/cliente.controller.js");  
  
  var router = require("express").Router();  
  
  // Crear un nuevo cliente  
  router.post("/", isAuthenticated, cliente.create);  
  
  // Obtener los clientes  
  router.get("/", isAuthenticated, cliente.findAll);  
  
  // Obtener un cliente con su id  
  router.get("/:id_cliente", isAuthenticated, cliente.findOne);  
  
  // Actualizar un cliente con su id  
  router.put("/:id_cliente", isAuthenticated, cliente.update);  
  
  // Borrar un cliente con su id  
  router.delete("/:id_cliente", isAuthenticated, cliente.delete);  
  
  // Borrar todos los clientes  
  router.delete("/", isAuthenticated, cliente.deleteAll);  
  
  app.use(`/${API}/${REST}/${VERSION}/cliente`, router);  
};
```

Si vamos al cliente podemos ver una de las funciones del controlador, la cual crea un nuevo cliente y devuelve un estado HTTP y un mensaje en funcion de lo que le devuelva la funcion del modelo.

```
const Cliente = require("../models/cliente.model.js");

// Crear un nuevo Cliente
exports.create = (req, res) => {
  // Validamos la petición
  if (!req.body) {
    res.status(400).send({
      message: "La petición no puede estar vacía!",
    });
  }
  // Creamos un Cliente
  const cliente = new Cliente({
    id_user: req.body.id_user,
    nombre: req.body.nombre,
    apellidos: req.body.apellidos,
    fecha_nacimiento: req.body.fecha_nacimiento,
    imagen: req.body.imagen,
  });

  // Guardar el Cliente en la DB
  Cliente.create(cliente, (err, data) => {
    if (err)
      res.status(500).send({
        message: err.message || "Un error ocurrió al crear el Cliente.",
      });
    else res.send(data);
  });
};
```

Por último en el modelo tenemos la consulta SQL

```
const conn = require("../db.js");

// constructor
const Cliente = function (cliente) {
  this.id_cliente = cliente.id_cliente;
  this.id_user = cliente.id_user;
  this.nombre = cliente.nombre;
  this.apellidos = cliente.apellidos;
  this.fecha_nacimiento = cliente.fecha_nacimiento;
  this.imagen = cliente.imagen;
};

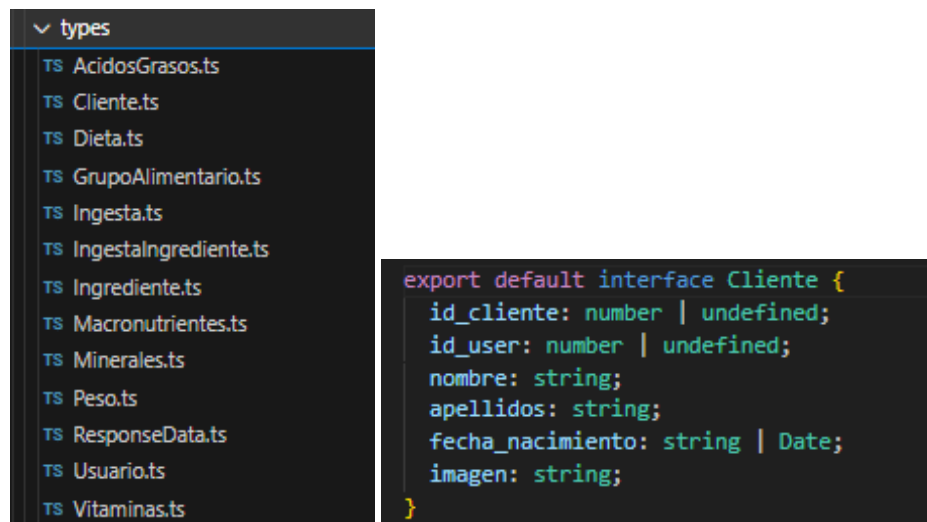
Cliente.create = (nuevoCliente, result) => {
  conn.query("INSERT INTO cliente SET ?", nuevoCliente, (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(err, null);
      return;
    }

    console.log("cliente creado: ", { id: res.insertId, ...nuevoCliente });
    result(null, { id: res.insertId, ...nuevoCliente });
  });
};
```

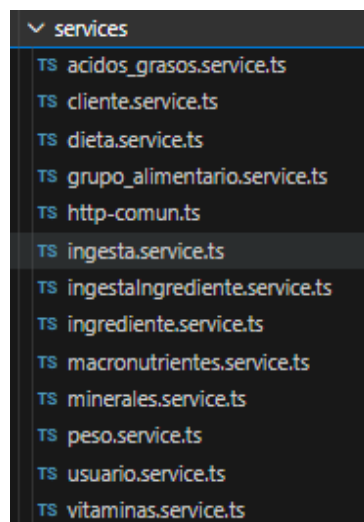
El resto de funciones de los ficheros siguen este mismo esquema variando ligeramente este esquema.

3.2.3 FrontEnd

En el FrontEnd volvemos a utilizar la arquitectura de MVC para recibir y mostrar los datos por pantalla. En este caso los modelos serían los types:



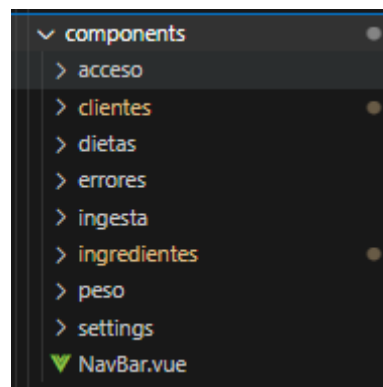
Los controladores serian los services:



```
import type Cliente from "@/types/Cliente";
import http from "@/http-comun";
import type ResponseData from "@/types/ResponseData";

class ClienteService {
  post(data: Cliente): Promise<any> {
    return http.post(`/cliente/`, data).then((response: ResponseData) => {
      return response.data;
    });
  }
  getAll(): Promise<any> {
    return http.get(`/cliente/`).then((response: ResponseData) => {
      return response.data;
    });
  }
  getOne(id_cliente: number): Promise<any> {
    return http.get(`/cliente/${id_cliente}`).then((response: ResponseData) => {
      return response.data;
    });
  }
  update(id_cliente: number, data: any): Promise<any> {
    return http
      .put(`/cliente/${id_cliente}`, data)
      .then((response: ResponseData) => {
        return response.data;
      });
  }
  delete(id_cliente: number): Promise<any> {
    return http
      .delete(`/cliente/${id_cliente}`)
      .then((response: ResponseData) => {
        return response.data;
      });
  }
}
```

Y las vistas serían los components:



Aquí vemos un ejemplo de llamada a un cliente dentro de un componente.

```

methods: {
  getCientes() {
    ClienteService.getAll().then((data) => {
      this.listaClientes = data;
    }).catch((err) => {
      const store = useMessageStore()
      if (err.response && err.response.status == 403) {
        store.message = "Necesitas estar logueado."
        return
      }
      store.message = !err.response ? GENERIC_ERR_MESSAGE : err.response.data.message
    });
  },
},

```

Como estamos utilizando VueJS hemos creado componentes que se comunican entre si a través de pasar “props” de padres a Hijos.

```

<div class="col-12">
  <GraficoPeso id_grafico="graficoPeso" :data-peso="dataGrafico"></GraficoPeso>
</div>

```

O de recibir información de hijos a padres recibiendo a través de eventos.

```

<!-- Modales -->
<Login @emit-logged="checkAuth"></Login>
<Registro></Registro>
</template>

```

Por ultimo para tener información disponible a traves de todo el frontend hemos usado una store creada con Pinia.

```

import { defineStore } from "pinia";

export const useMessageStore = defineStore("messages", {
  state: () => ({ message: "" }),
  actions: {
    setMessage(newMessage: string) {
      this.message = newMessage;
    },
  },
});

```

3.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

Para el despliegue hemos escogido empaquetar todo con Docker para simplificar la puesta en marcha, haciendo que solamente tengamos que ejecutar el comando:

docker compose build

Y luego el comando:

docker compose up

Para iniciar el despliegue.

Los servicios que tenemos en nuestro fichero de docker compose son:

- La base de datos la cual tiene un servidor de MySQL.
- El Backend el cual tiene un servidor de NodeJS.
- El Frontend el cual tiene un servidor de NGinx.

4 Planificación

Las fases del proyecto que nos hemos marcado junto con el tiempo que nos ha llevado cada apartado.

- Análisis (3h)
- Diseño (20h)
- Desarrollo (92h total)
 - Base de Datos (4h)
 - BackEnd (8h)
 - FrontEnd (60h)
 - Corrección de Bugs (20h)
- Compilación (4h)
- Despliegue (5h)

El tiempo total estimado ascendería a 124 horas.

5 Documentación Técnica: análisis, diseño, implementación y pruebas.

5.1 Especificación de requisitos

Debido a que esta es una aplicación web podemos vamos a diferenciar entre los requisitos del servidor y los requisitos del cliente. En el caso del servidor, debido a que hemos empaquetado la aplicación con Docker para funcionar con Docker Compose necesitaríamos los requisitos de Docker.

En el servidor:

- WSL versión 1.1.3.0 o superior.
- Windows 10 64-bit 21H2 (build 19044) o superior
- Procesador de 64 Bits.
- BIOS que acepte virtualización
- 8 GB de RAM o más.

En el cliente:

- Cualquier navegador web que acepte JavaScript ES6, HTML5 y CSS

5.2 *Análisis del sistema*

¿????

5.3 *Diseño del sistema:*

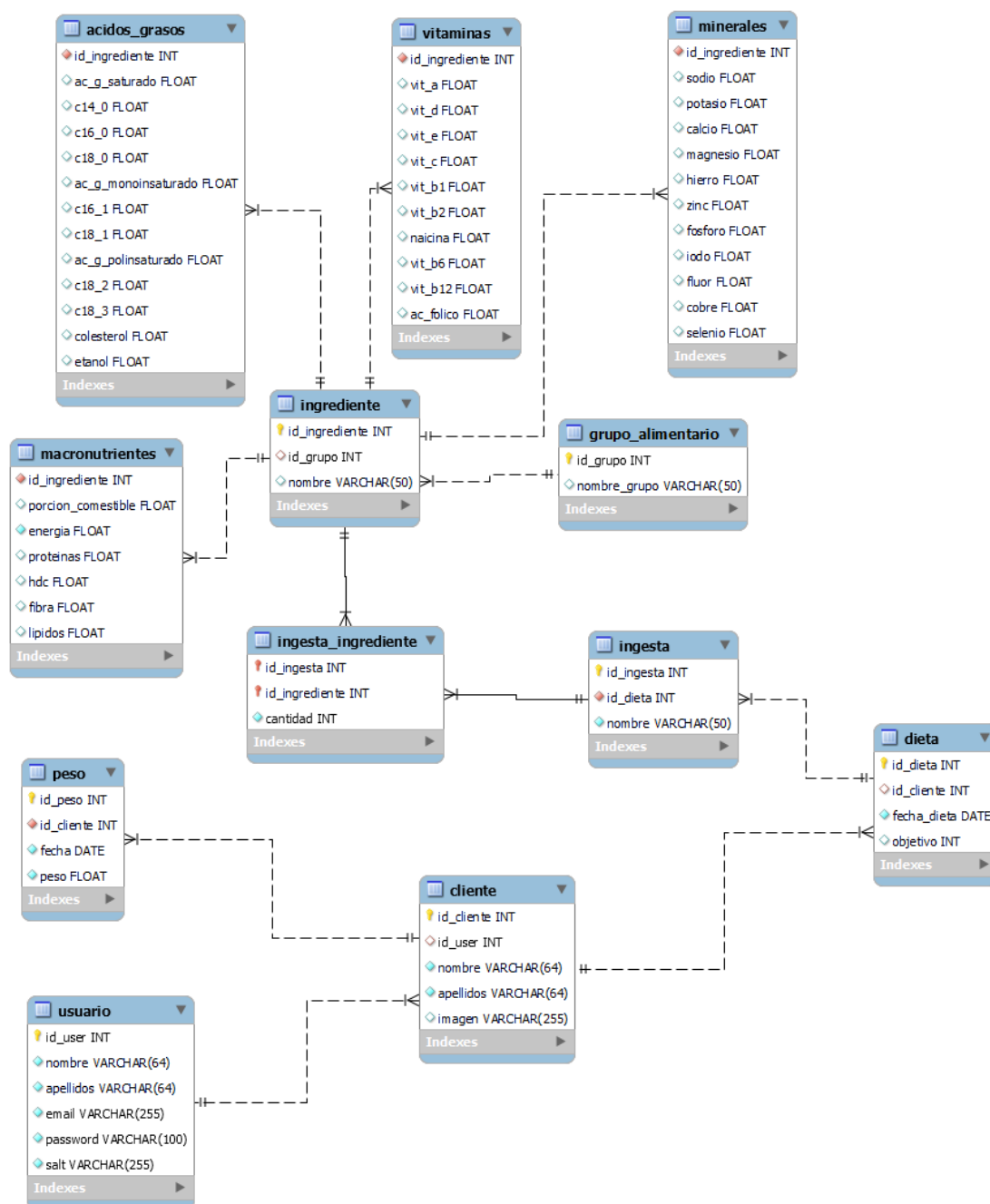
5.3.1 *Diseño de la Base de Datos*

La base de datos tiene un total de 12 tablas, las cuales son:

- Usuario
- Cliente
- Peso
- Dieta
- Ingesta
- Ingesta Ingrediente
- Ingrediente
- Grupo Alimentario
- Macronutrientes
- Ácidos Grasos
- Vitaminas
- Minerales

La tabla central de todo el proyecto sería el cliente, puesto que toda la mayoría de las funcionalidades que se vayan a implementar van a girar en torno a los clientes.

El diagrama entidad relación podemos verlo a continuación.

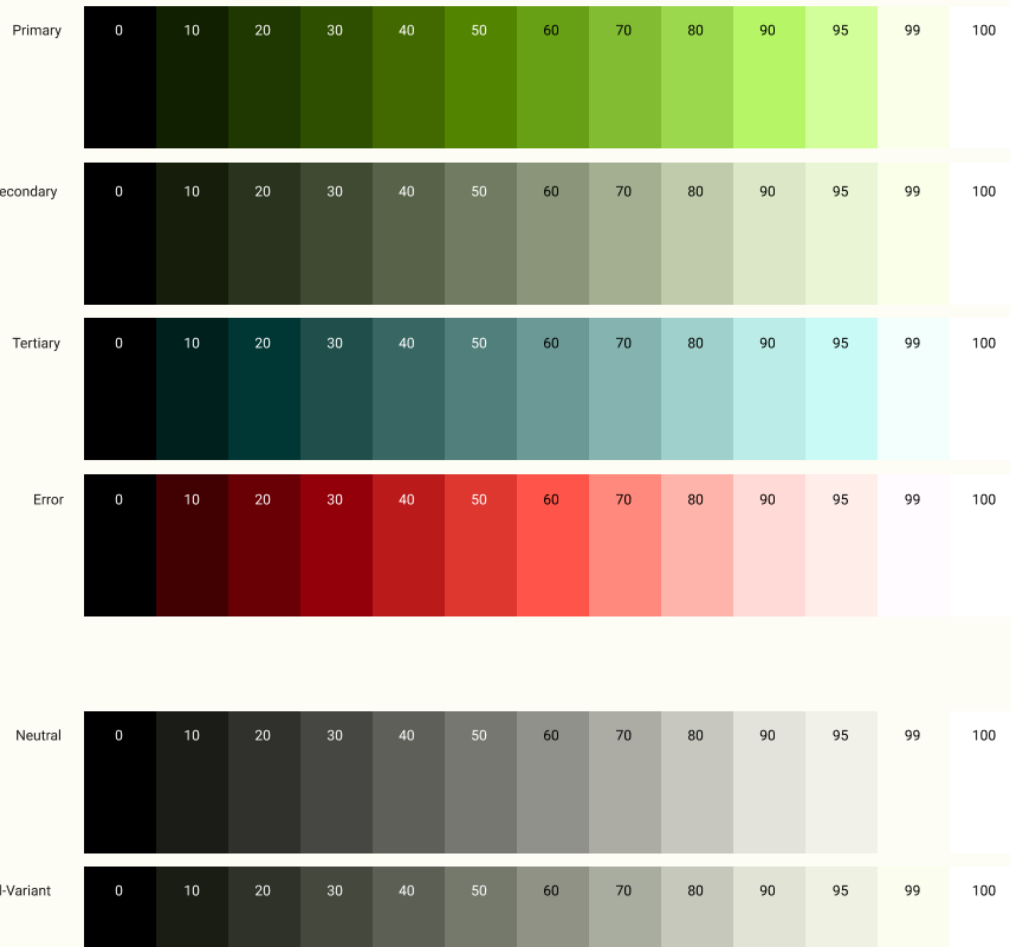


5.3.2 Diseño de la Interfaz de usuario.

Después de crear el prototipo con Figma esta es la interfaz que hemos propuesto:

El esquema de color escogido:

Tonal Palettes



Light Scheme



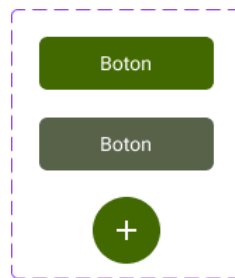
Dark Scheme



Light Surfaces	Dark Surfaces
Surface	Surface
Surface at +1	Surface at +1
+ 5% Primary	+ 5% Primary
Surface at +2	Surface at +2
+ 8% Primary	+ 8% Primary
Surface at +3	Surface at +3
+ 11% Primary	+ 11% Primary
Surface at +4	Surface at +4
+ 12% Primary	+ 12% Primary
Surface at +5	Surface at +5
+ 14% Primary	+ 14% Primary

Todos los componentes utilizados:

- Botones



- Gráficos



- Elementos del acordeón

Arroz	18 kcal
80 gr	15/3/0

- Acordeón

Desayuno	200 Kcal	▼
Almuerzo	200 Kcal	▼
Comida	200 Kcal	▼
Merienda	200 Kcal	▼
Cena	200 Kcal	▼

Desayuno	200 Kcal	▲
200 Kcal		
100/50/50		
Arroz	18 kcal	
80 gr	15/3/0	
Arroz	18 kcal	
80 gr	15/3/0	
Arroz	18 kcal	
80 gr	15/3/0	
+		
Almuerzo	200 Kcal	▼
Comida	200 Kcal	▼
Merienda	200 Kcal	▼
Cena	200 Kcal	▼

- SideMenu

Cientes

Usuario 1	>
Usuario 2	>
Usuario 3	>

+

Cientes

Usuario 1	>
Usuario 2	>
Usuario 3	>

+

- Login



Email

Contraseña

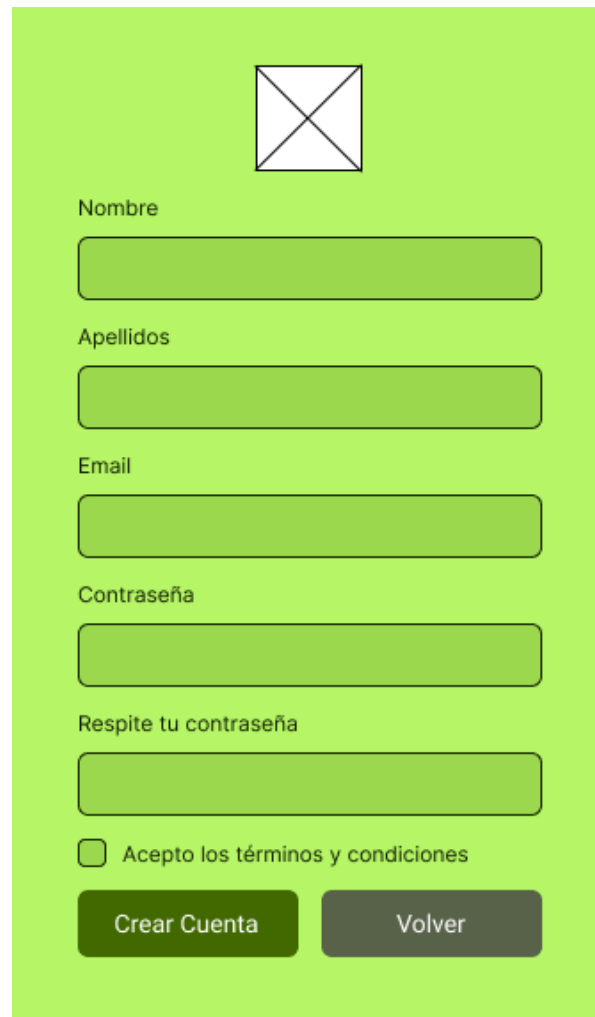
He olvidado mi contraseña

☐ Recordarme

Iniciar Sesion

Registrarse

- Registro



Nombre

Apellidos

Email

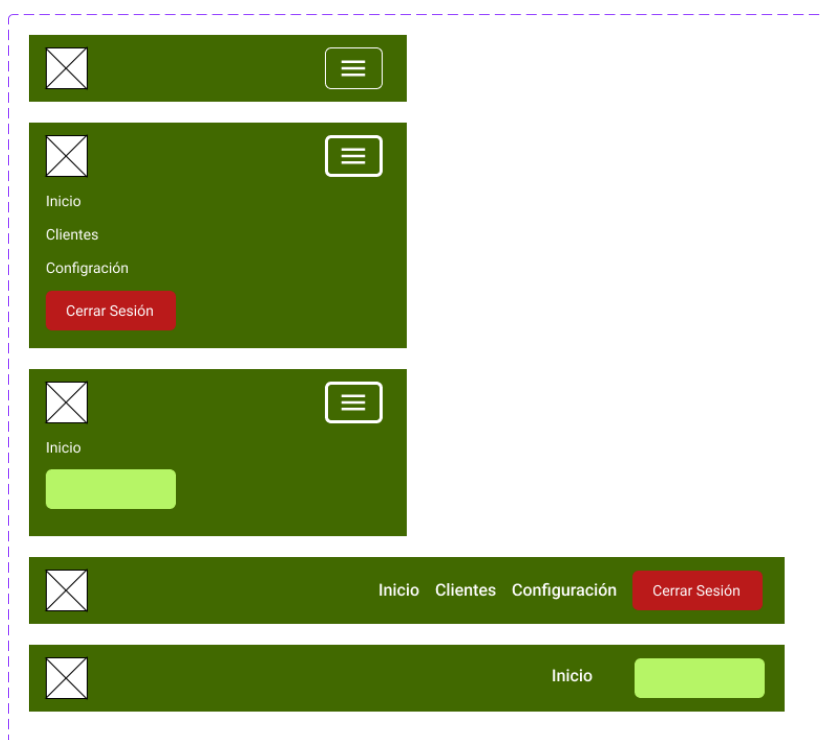
Contraseña

Respite tu contraseña

☐ Acepto los términos y condiciones

Crear Cuenta Volver

- NavBar



Una vez todos los componentes combinados tenemos:

- Versión para móvil



- Versión para sobremesa



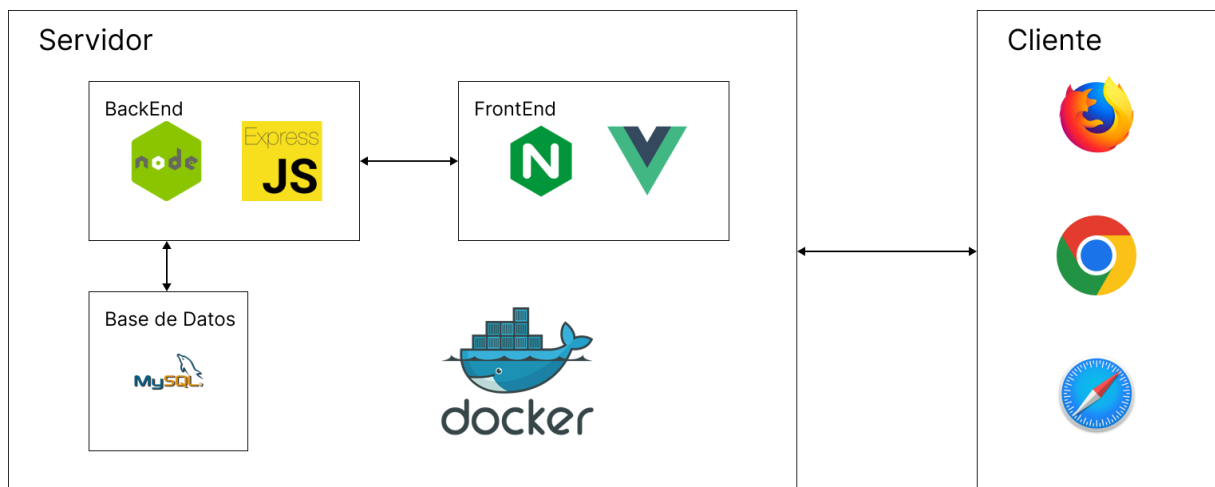
Con este prototipo hemos buscado definir las funcionalidades básicas que hemos implementado en esta iteración del proyecto. En este caso hemos diseñado las siguientes funcionalidades.

- Acceso a la aplicación
- Registro en la aplicación
- Acceso y modificación de los datos usuario

- Listado y creación de clientes del usuario.
- Ver datos del cliente y editarlos
- Ver dietas y modificarlas
- Añadir y modificar ingestas

5.3.3 Diseño de la Aplicación.

La aplicación tiene el siguiente diseño a la hora de comunicarse entre servidor y cliente.



5.4 Implementación:

5.4.1 Entorno de desarrollo.

Como entorno de desarrollo (IDE) hemos utilizado Visual Studio Code.

5.5 Pruebas.

No hemos llegado a implementar pruebas ni unitarias ni de integración.

6 Manuales de usuario

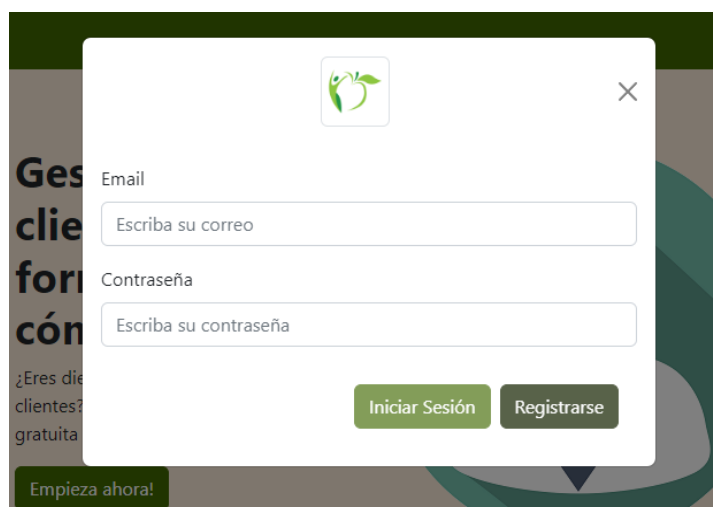
6.1 Manual de usuario

6.1.1 Acceso

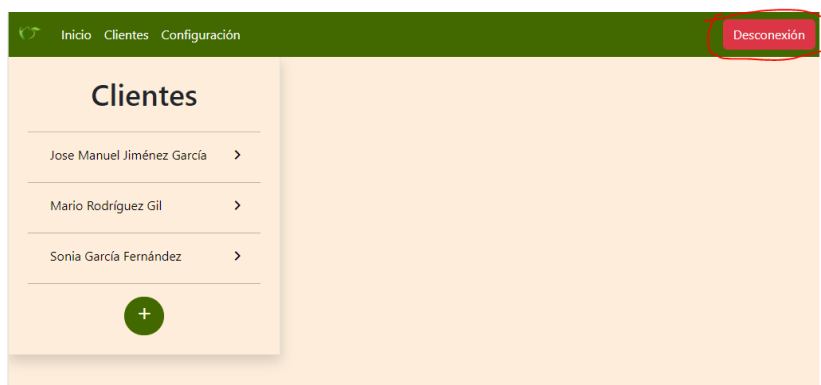
Pulsamos el botón Acceso en la barra de navegación.



Si tenemos una cuenta creada en la aplicación escribimos nuestro correo y nuestra contraseña y pulsamos el boton iniciar sesión.

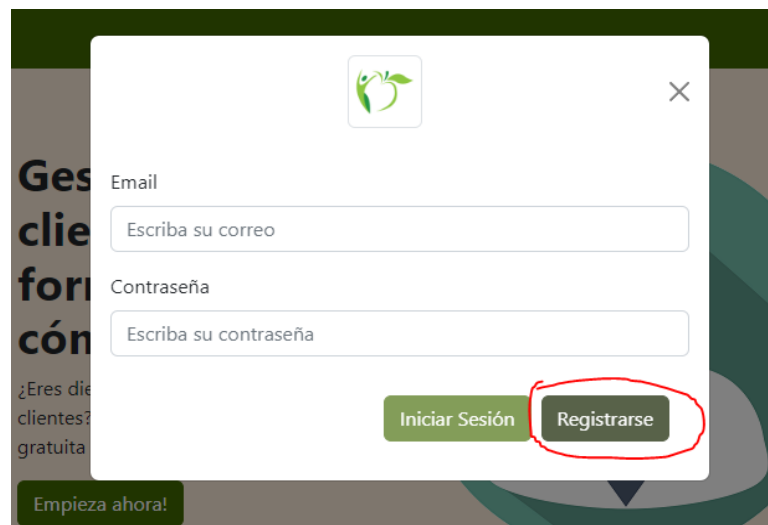


Podemos cerrar sesión pulsando el botón desconexión de la barra de navegación.



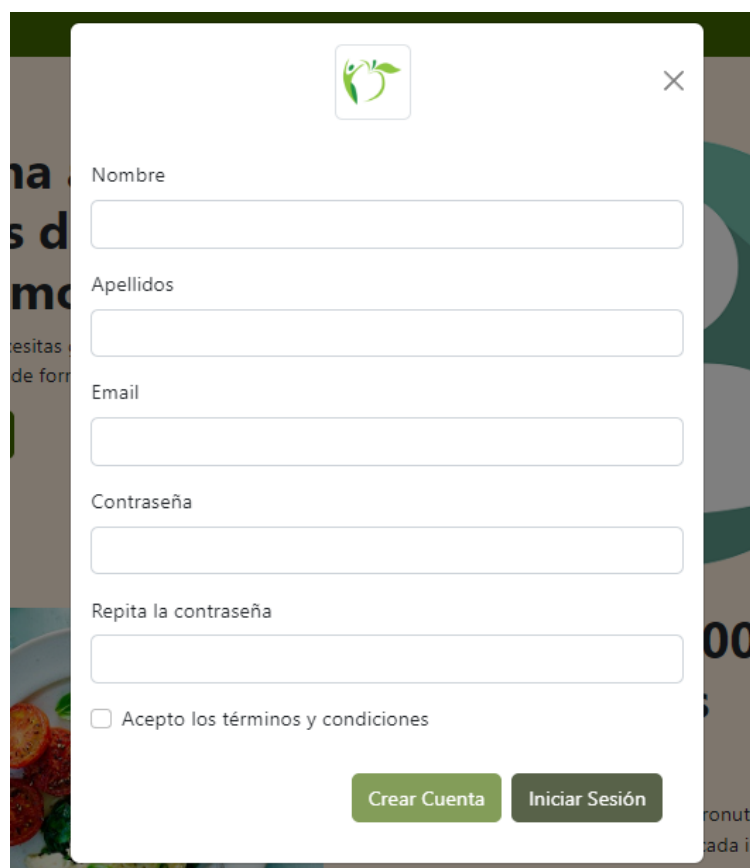
6.1.2 Registro

Podemos acceder al registro desde el Acceso, pulsando el botón Registrarse.



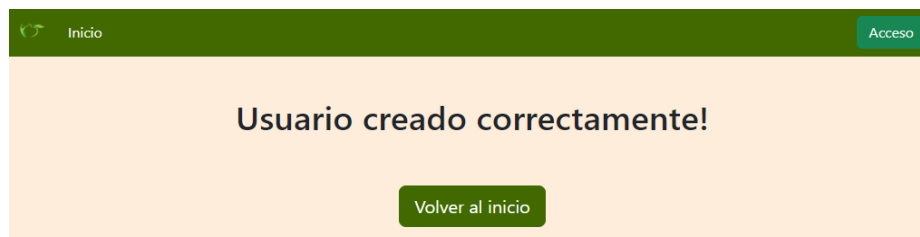
A modal window with a green apple icon and a close button. It contains two input fields: 'Email' with the placeholder 'Escriba su correo' and 'Contraseña' with the placeholder 'Escriba su contraseña'. Below these fields are two buttons: 'Iniciar Sesión' and 'Registrarse'. The 'Registrarse' button is circled in red. At the bottom left of the modal, there is a green button labeled 'Empieza ahora!'.

Ahora podemos acceder a nuestra cuenta pulsando el boton Registrarse. Y rellenando todos los campos.



A modal window with a green apple icon and a close button. It contains five input fields: 'Nombre', 'Apellidos', 'Email', 'Contraseña', and 'Repita la contraseña'. Below these fields is a checkbox labeled 'Acepto los términos y condiciones'. At the bottom right of the modal are two buttons: 'Crear Cuenta' and 'Iniciar Sesión'.

Una vez registrado nos mostrará el siguiente mensaje.

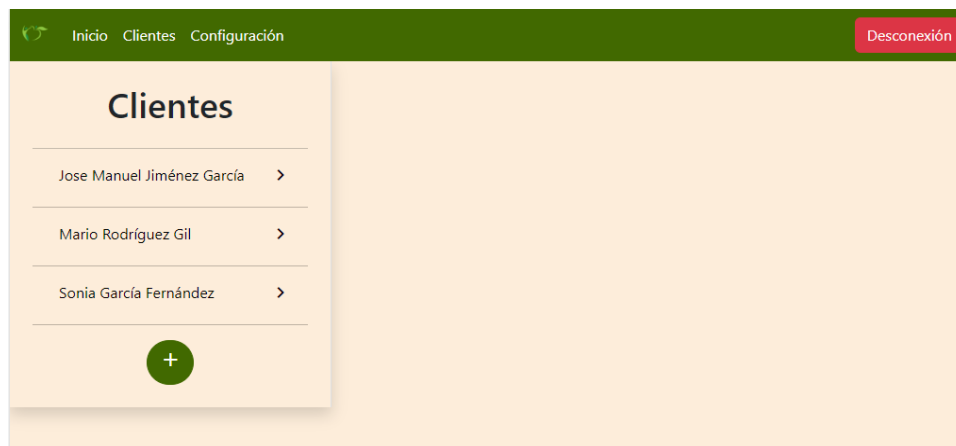


6.1.3 Clientes

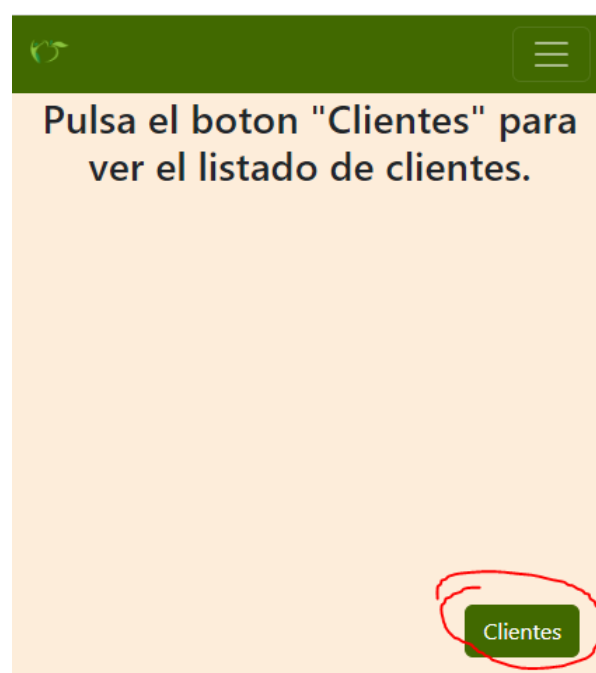
6.1.3.1 Listado de clientes

Al acceder a la aplicación nos mostrará la lista de clientes que tengamos, en caso de no tener ninguno, nos lo mostrará vacío.

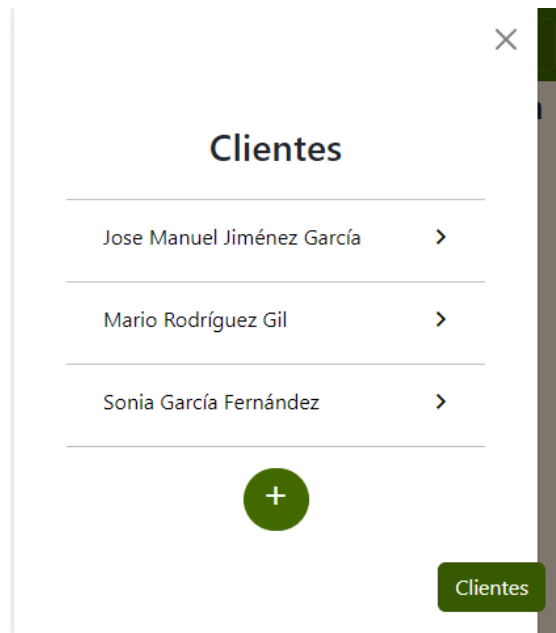
En la versión sobremesa lo veremos así.



Y en la versión móvil tendremos que pulsar el botón clientes para poder ver la lista.

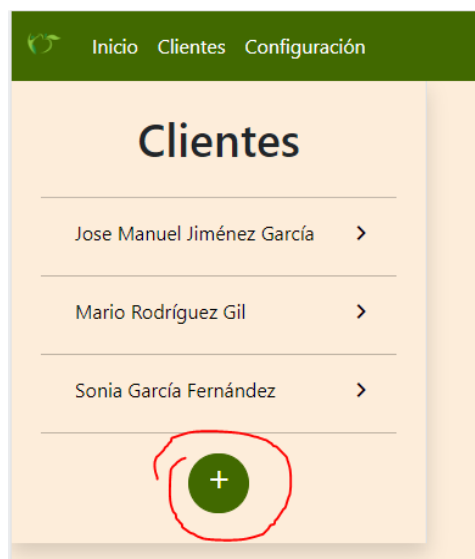


Al pulsarlo nos mostrará la lista en un menu lateral.

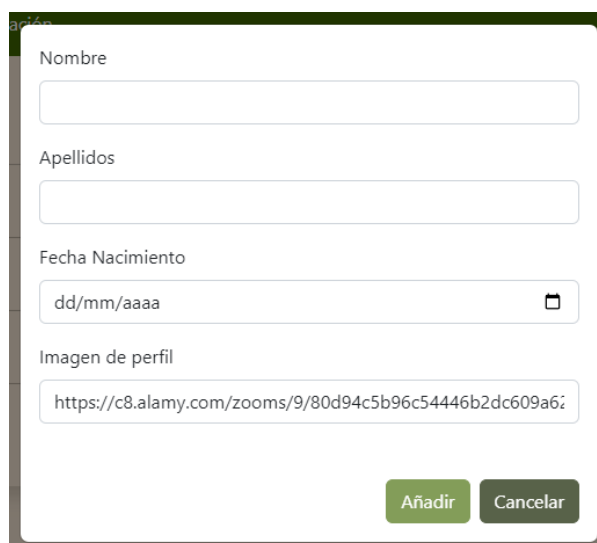


6.1.3.2 Añadir cliente


Para añadir un cliente tenemos que pulsar el botón “+”.



Rellenamos los datos del cliente y le damos a añadir. Actualmente la imagen del perfil tiene por defecto una imagen generica, en el futuro esta parte cambiará.



Formulario para añadir un cliente:

- Nombre:
- Apellidos:
- Fecha Nacimiento: 
- Imagen de perfil:
- Botones: **Añadir** (verde) y **Cancelar** (gris)

6.1.3.3 Eliminar un cliente

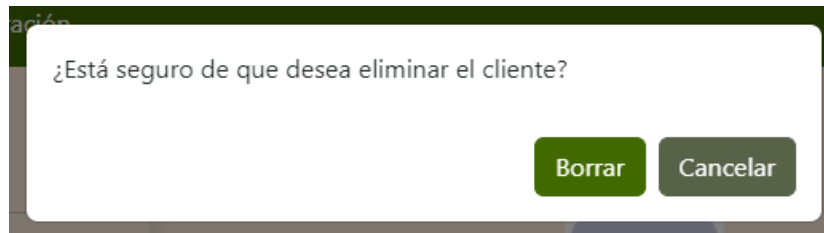
Seleccionamos el cliente que queremos eliminar pulsando sobre su nombre.



Pulsamos el boton de borrar.



Nos pedirá confirmar la operación.




6.1.3.4 Editar un cliente

Simplemente pulsamos sobre cualquiera de los datos del cliente y nos mostrará un formulario para editar al cliente.



Modificamos los datos que deseemos cambiar y pulsamos el botón Actualizar.



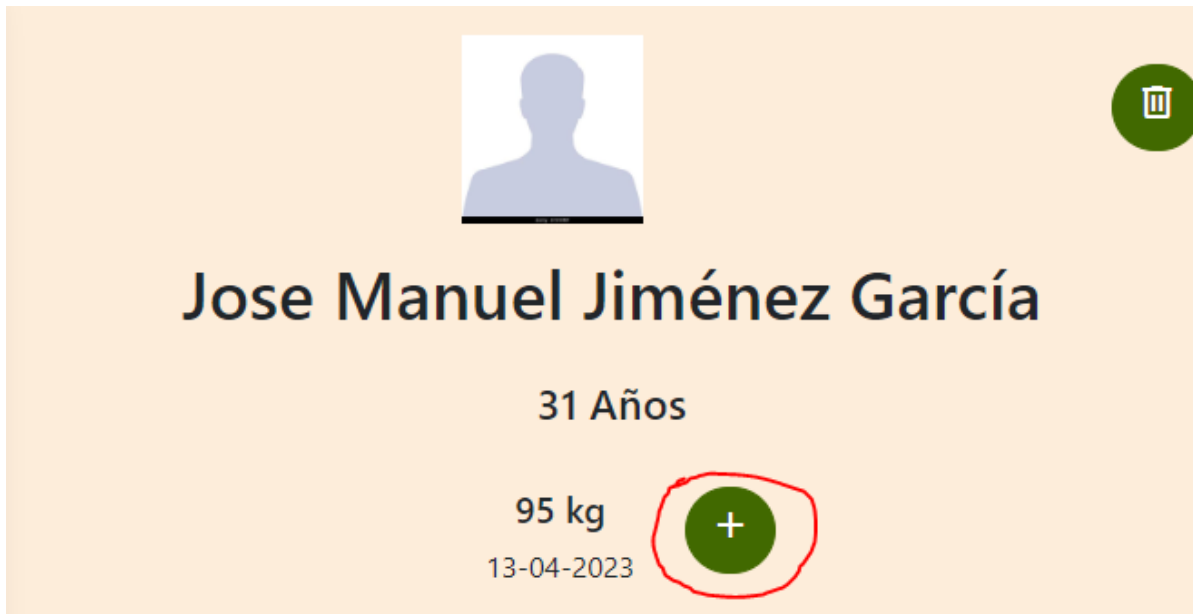
Formulario de edición de perfil con los siguientes campos:

- Nombre: Jose Manuel
- Apellidos: Jiménez García
- Fecha Nacimiento: 26/02/1992
- Imagen de perfil: <https://c8.alamy.com/zooms/9/80d94c5b96c54446b2dc609a62>

Botones: Actualizar, Cancelar

6.1.3.5 Añadir un peso

Pulsamos el botón “+” al lado del peso.



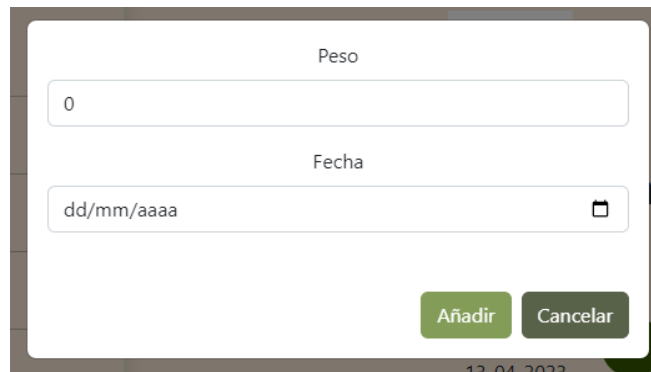
Perfil de Jose Manuel Jiménez García, 31 Años.

Peso: 95 kg

Fecha: 13-04-2023

Botón de añadir (+) circulado en rojo.

Rellenamos el peso y la fecha del peso y pulsamos el botón Añadir.



Peso

0

Fecha

dd/mm/aaaa

Añadir Cancelar

Automaticamente nos actualizará el grafico con el peso.



6.1.4 Dietas

6.1.4.1 Añadir una dieta

Pulsamos el botón añadir al lado de las dietas.



Dietas

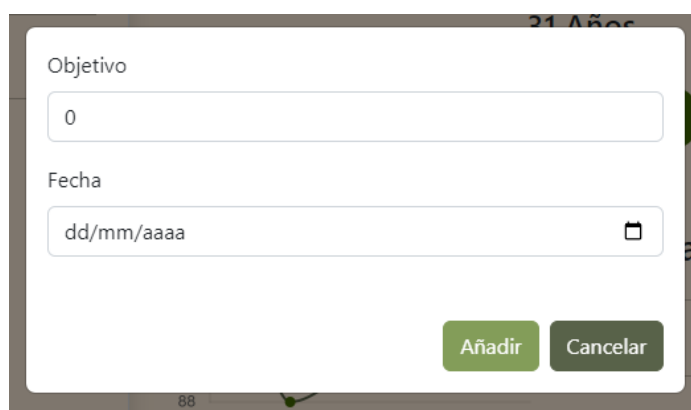
07-04-2023

06-04-2023

05-04-2023

+

Indicamos las Kcal objetivo para esa dieta y la fecha.



Objetivo

0

Fecha

dd/mm/aaaa

Añadir Cancelar

6.1.4.2 Consultar una dieta

Seleccionamos la fecha de la dieta que queremos consultar.



Dietas

07-04-2023

06-04-2023

05-04-2023

+

Ahora veremos el cliente y la fecha para el cual es la dieta, cuantas Kcal faltan para pasarnos del objetivo, los porcentajes de Grasas, Hidratos de Carbono y Proteínas que tiene nuestra dieta y por último veremos cada una de las ingestas que le hemos creado al usuario y las Kcal asignadas a dicha ingesta.



Jose Manuel

5/4/2023

Objetivo del día

1500 - 686 = 814

Grasas HdC Proteínas

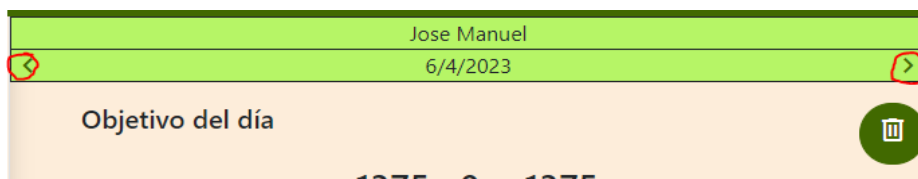
desayuno 686 Kcal

almuerzo

comida

6.1.4.3 Alternar entre dietas

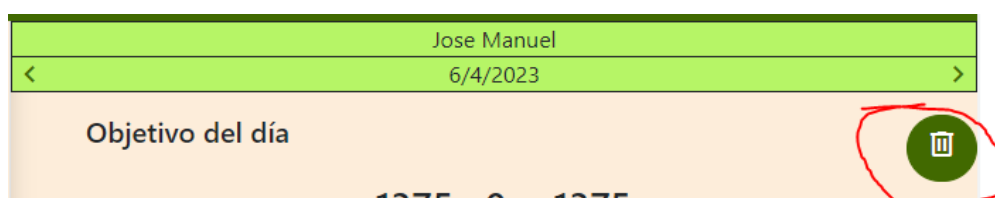
Para alternar entre las distintas dietas creada solo debemos pulsar los siguientes botones.



The screenshot shows a header bar with a green background. The top bar contains the name "Jose Manuel". The second bar contains the date "6/4/2023". On the left side of the second bar, there is a green circle with a white left-pointing arrow. On the right side, there is a green circle with a white right-pointing arrow. Below the header, the text "Objetivo del día" is visible. At the bottom, the numbers "1275 0 1275" are partially visible.

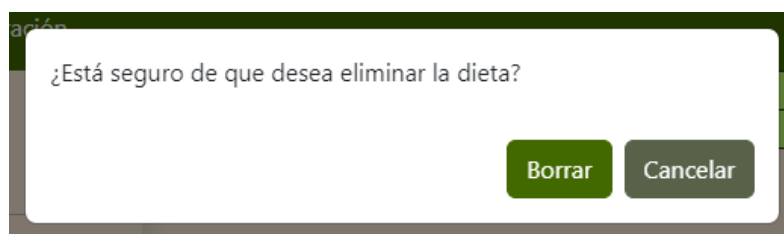
6.1.4.4 Eliminar una dieta

Para eliminar una dieta pulsaremos el botón de eliminar.



This screenshot is similar to the previous one, showing the header with "Jose Manuel" and "6/4/2023". The "Objetivo del día" section is visible. A green circular button with a white trash icon is located on the right side of the "Objetivo del día" section and is circled in red.

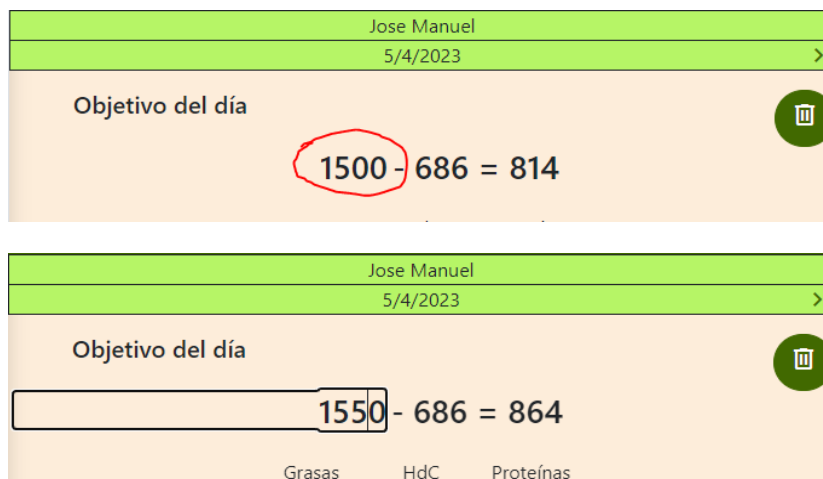
Nos pedirá confirmación antes de borrar.



A white dialog box with a green border is shown. It contains the text "¿Está seguro de que desea eliminar la dieta?". At the bottom right, there are two buttons: "Borrar" (green) and "Cancelar" (grey).

6.1.4.5 Modificar una dieta

De una dieta únicamente podemos modificar el objetivo, para ello pulsaremos sobre el y lo modificaremos. Al pinchar fuera del campo, se actualizará.

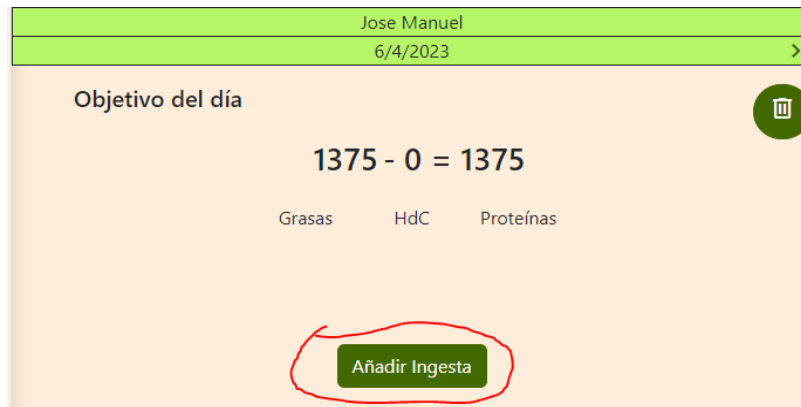


Two screenshots are shown. The top one shows the "Objetivo del día" section with the calculation "1500 - 686 = 814". The number "1500" is circled in red. The bottom screenshot shows the same section, but the number "1500" has been changed to "1550" in a text input field, resulting in the calculation "1550 - 686 = 864". Below the calculation, the labels "Grasas", "HdC", and "Proteínas" are visible.

6.1.5 Ingestas

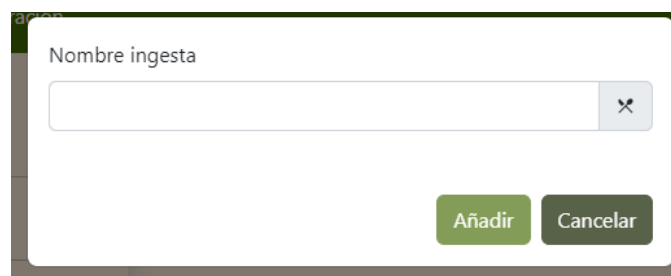
6.1.5.1 Añadir una ingesta

Para añadir una ingesta pulsaremos el botón Añadir Ingesta



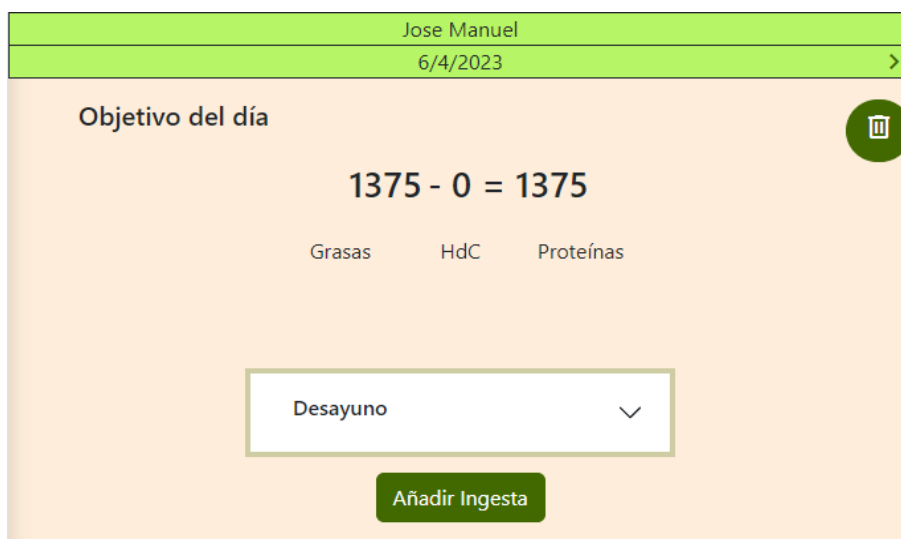
The screenshot shows the top part of the application interface. At the top, there is a green header bar with the text 'Jose Manuel' and '6/4/2023'. Below this, the main area has a light orange background. It displays 'Objetivo del día' followed by the calculation '1375 - 0 = 1375'. Underneath the calculation are three labels: 'Grasas', 'HdC', and 'Proteínas'. At the bottom center, there is a green button labeled 'Añadir Ingesta' which is circled in red. To the right of the main content area, there is a green circular icon with a white trash can symbol.

Le damos un nombre a la ingesta



The screenshot shows a modal dialog box titled 'Nombre ingesta'. It contains a text input field with a small 'x' icon on the right. Below the input field are two buttons: 'Añadir' (green) and 'Cancelar' (grey).

Y nos la añadirá a la dieta.



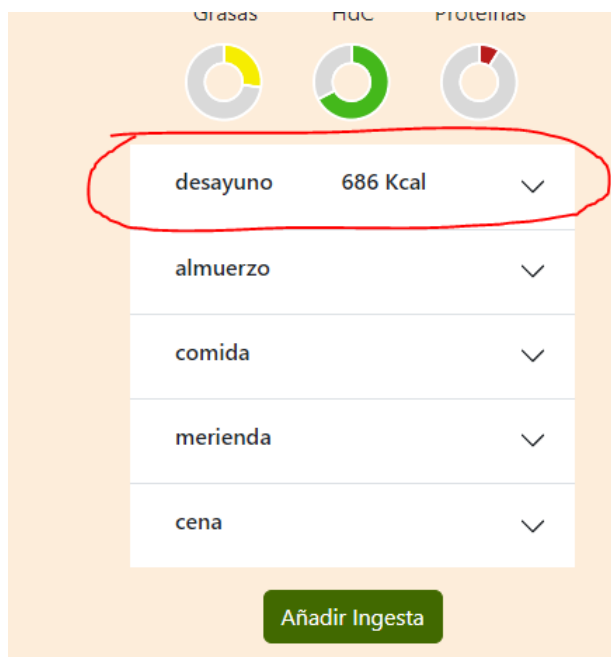
The screenshot shows the same interface as before, but now with an additional input field. Below the 'Grasas', 'HdC', and 'Proteínas' labels, there is a white dropdown menu with the text 'Desayuno' and a downward arrow. Below this dropdown is a green button labeled 'Añadir Ingesta'.

En las ingestas cuando tengamos ingredientes veremos siempre, las Kcal totales de la ingesta, y 3 numeros que representan las Grasas, los Hidratos de Carobono y las Proteínas.

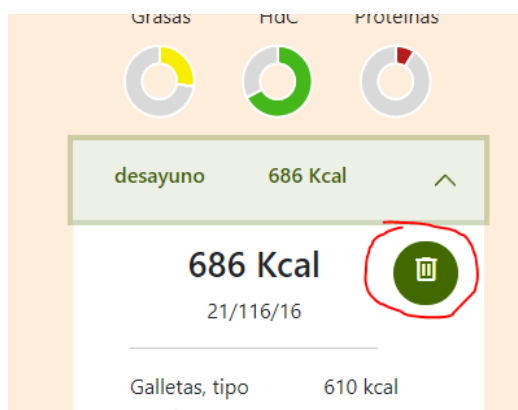


6.1.5.2 Borrar una ingesta

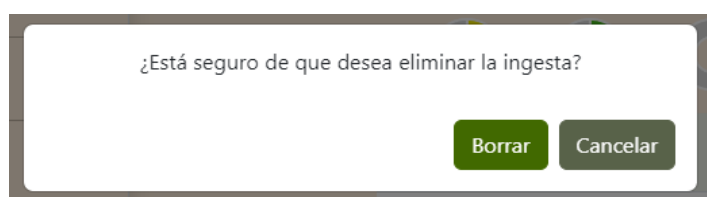
Desplegamos la ingesta que queremos eliminar pulsando sobre ella.



Pulsamos el boton Eliminar.



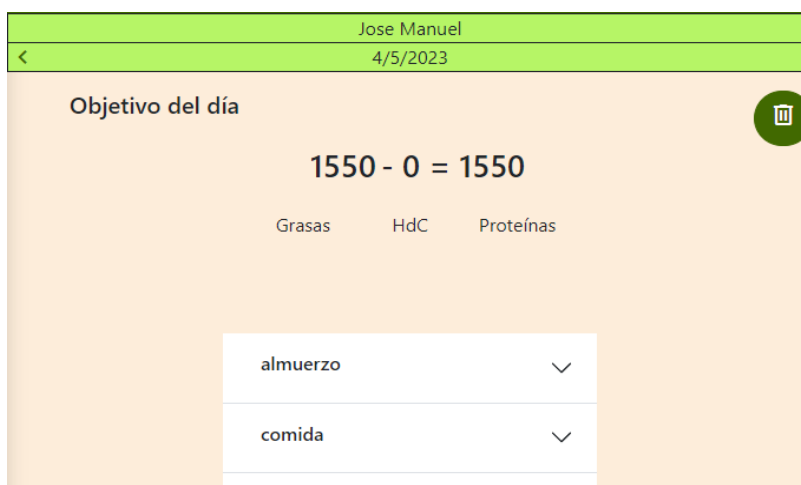
Nos pedirá confirmación antes de borrar.



¿Está seguro de que desea eliminar la ingesta?

Borrar Cancelar

Una vez eliminado nos actualizará la pantalla.



Jose Manuel

< 4/5/2023

Objetivo del día

1550 - 0 = 1550

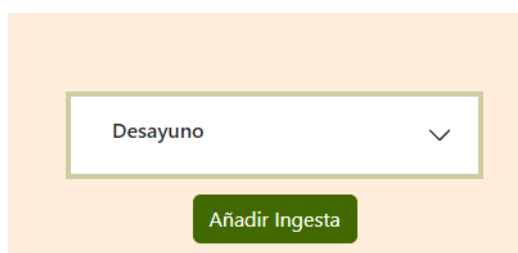
Grasas HDL Proteínas

almuerzo ^

comida ^

6.1.5.3 Añadir un ingrediente a una ingesta

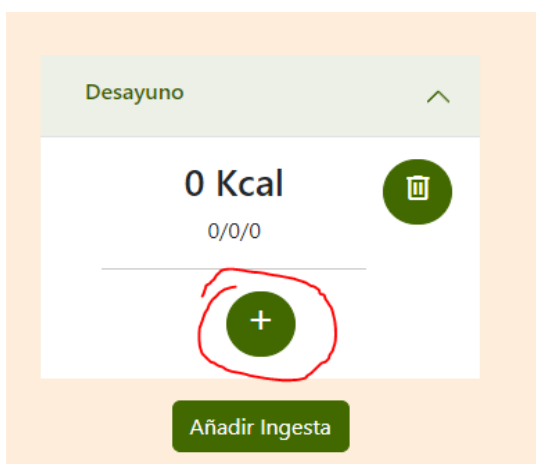
Desplegamos la ingesta a la cual queremos añadirle el ingrediente



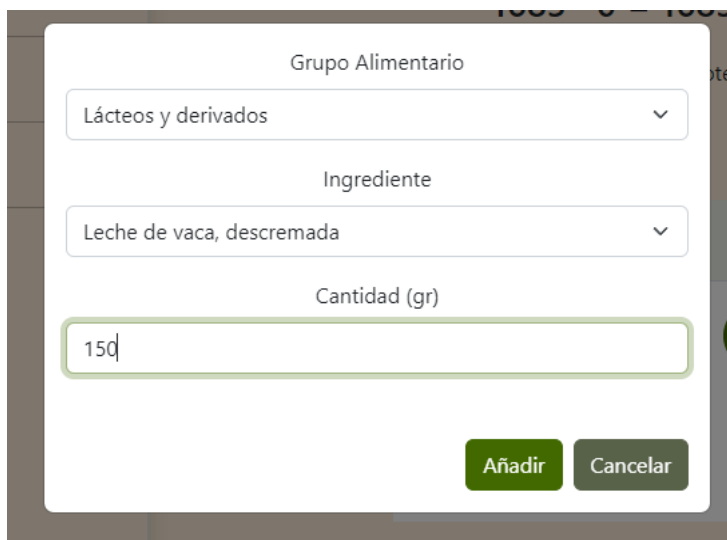
Desayuno ^

Añadir Ingesta

Pulsamos el botón "+".



Nos mostrará un formulario en el cual podemos seleccionar un ingrediente de la lista. También podemos filtrar la lista de ingredientes seleccionando un grupo alimentario, si no seleccionamos ningún grupo nos mostrará la lista con todos los ingredientes.



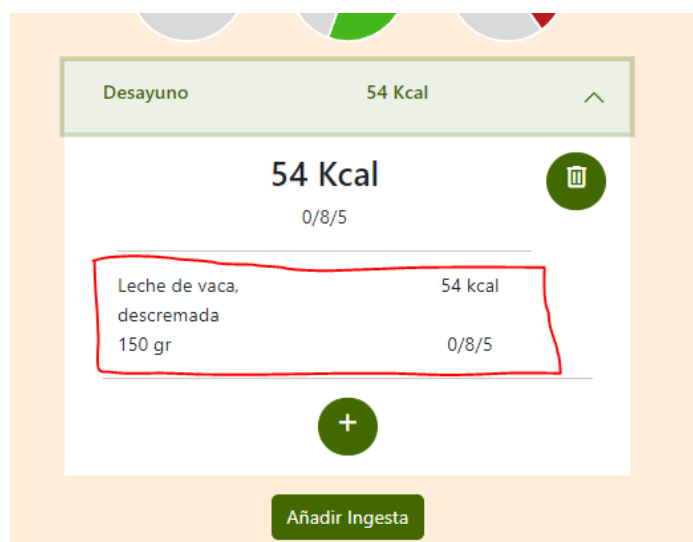
Una vez rellenados los campos ingrediente y cantidad podemos pulsar el botón añadir.



Y nos actualizará la información de la ingesta mostrándonos el nombre del ingrediente, la cantidad que hemos indicado, las Kcal totales, y las grasas, hidratos de carbono y proteínas de dicho ingrediente.

6.1.5.4 Modificar un ingrediente de una ingesta

Si pulsamos sobre un ingrediente nos mostrará el modal de modificación del ingrediente.



En este modal podemos ver el nombre del ingrediente, la cantidad y la cantidad de grasas, hidratos de carbono y proteínas del ingrediente.



Leche de vaca, descremada

54 kcal

150 gr

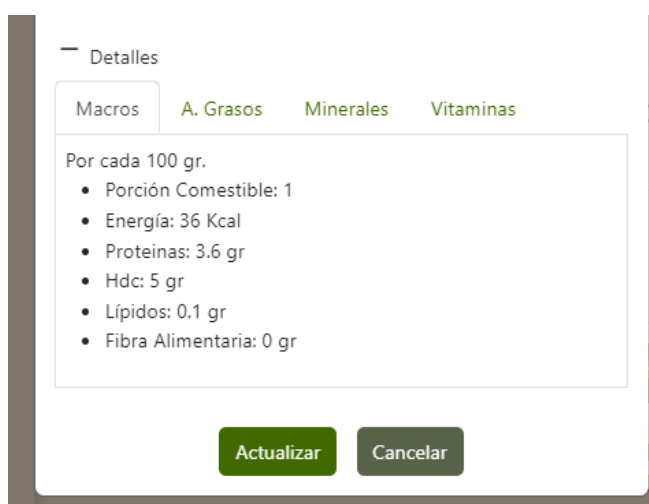
Grases HdC Proteínas

0 gr 8 gr 5 gr

+ Detalles

Actualizar Cancelar

Si pulsamos sobre más detalles nos mostrará la información sobre Macronutrientes, Ácidos Grasos, Minerales o Vitaminas.



Detalles

Macros A. Grasos Minerales Vitaminas

Por cada 100 gr.

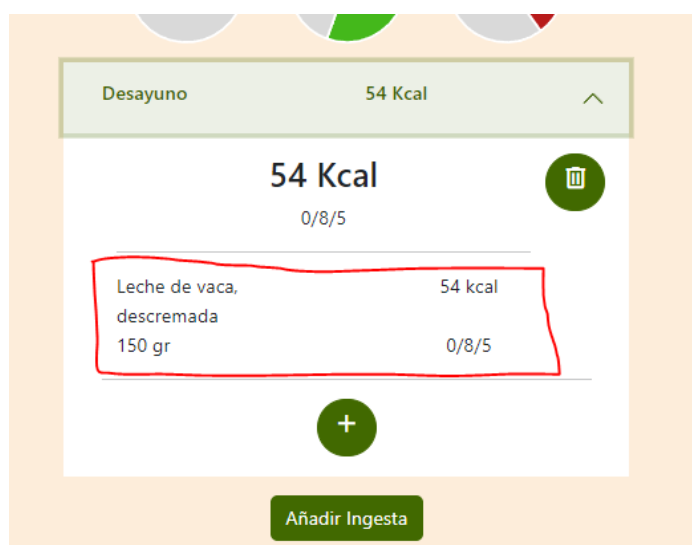
- Porción Comestible: 1
- Energía: 36 Kcal
- Proteínas: 3.6 gr
- HdC: 5 gr
- Lípidos: 0.1 gr
- Fibra Alimentaria: 0 gr

Actualizar Cancelar

Podemos modificar la cantidad de un ingrediente, una vez modificada podemos actualizarla pulsando el botón Actualizar. Esto hará que se actualice la dieta entera con la nueva información.

6.1.5.5 Eliminar un ingrediente de una ingesta

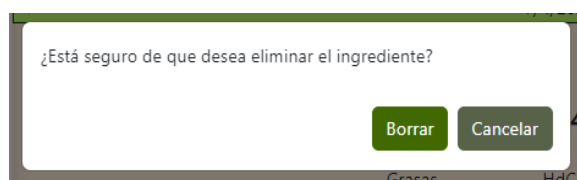
Si pulsamos sobre un ingrediente nos mostrará el modal de modificación del ingrediente.



Pulsamos sobre el botón de eliminar un ingrediente.



Y nos pedirá confirmación.



Una vez eliminado nos actualizará la información de la dieta y la ingesta.




6.1.6 Configuración

Si pulsamos sobre el enlace Configuración de la barra de navegación accederemos a la información del usuario.



En esta vista podemos ver y modificar los datos del usuario e incluso eliminar la cuenta.



6.1.6.1 Cambiar datos del usuario

Podemos modificar el nombre, los apellidos o el correo del usuario pulsando el botón Modificar.



Hola, alberto

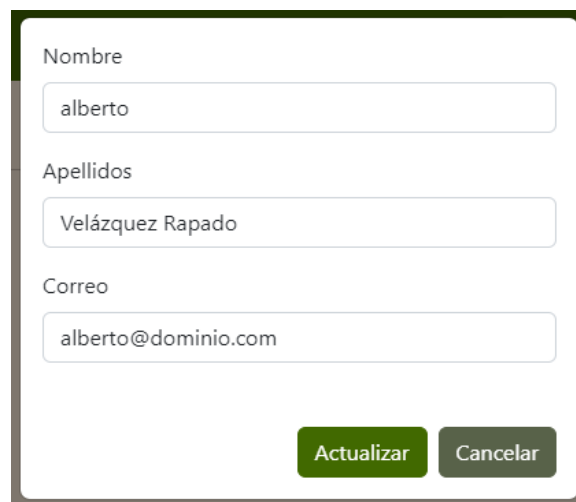
Nombre
alberto

Apellidos
Velázquez Rapado

Correo
alberto@dominio.com

Modificar

Esto nos mostrará un formulario con los datos actuales, una vez modificados pulsaremos el botón modificar.



Nombre

Apellidos

Correo

Actualizar **Cancelar**

Una vez modificados nos mostrará un mensaje indicándonos que los datos se han actualizado.

Usuario actualizado

Hola, Alberto

Nombre
Alberto

Apellidos
Velázquez Rapado

Correo
alberto@dominio.com

Modificar

6.1.6.2 Cambiar Contraseña

Para cambiar la contraseña pulsaremos el botón Cambiar Contraseña.

Contraseña

Cambiar contraseña

Este formulario nos pedirá la contraseña actual y la nueva contraseña.

Contraseña Actual

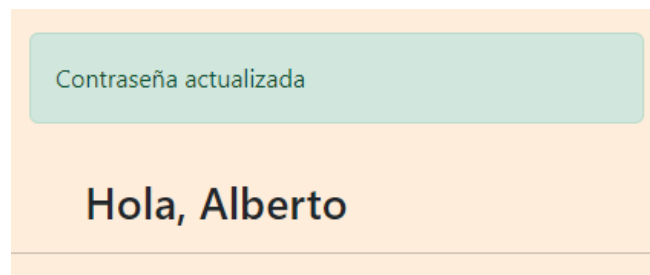
Nueva contraseña

Repita la contraseña

Cambiar Cancelar

Cambiar contraseña

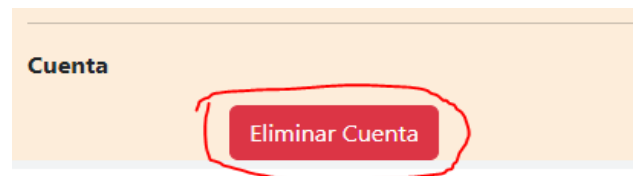
Una vez modificado nos mostrará un mensaje indicando que la contraseña ha sido actualizada.



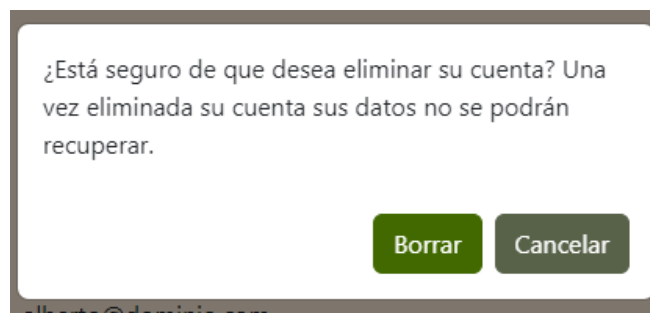
6.1.6.3 Eliminar cuenta

Es importante recordar que una vez se elimine la cuenta no se podrá recuperar nada de la información del usuario, por lo que se recomienda tener cuidado.

Para eliminar nuestra cuenta debemos pulsar el botón eliminar cuenta.



Nos pedirá confirmación.



Y si pulsamos sobre Borrar, borrará toda nuestra información de la base de datos y nos redireccionará a la pagina de inicio.



6.2 Manual de instalación

- Instalar Docker siguiendo los pasos que nos indique en la siguiente ruta:
<https://www.docker.com/>
- Modificar las variables de entorno del archivo ".env".
 - Modificar la variable MYSQLDB_ROOT_PASSWORD por la contraseña que deseemos.
 - Modificar la variable JWT_SECRET por el secreto que deseemos usar para encriptar el Token.
- Ejecutar el comando docker compose build para construir la aplicación
- Ejecutar el comando docker compose up para hacerlo funcionar.

7 Conclusiones y posibles ampliaciones

Las conclusiones que hemos sacado de este proyecto son:

- Desarrollar una aplicación completa es una tarea compleja que lleva mucho tiempo e implica tomar decisiones complejas.
- Trabajar con tecnologías por primera vez supone ampliar los tiempos de desarrollo y aumenta la probabilidad de error.
- La separación del backend y el frontend son fundamentales en las aplicaciones modernas.

Posibles ampliaciones:

- Añadir la posibilidad añadir elementos a consultar del cliente, como el somatotipo, el porcentaje de grasa corporal, medidas corporales, IMC, etc.
- Abrir la aplicación a que también haya otros roles de usuario en la aplicación que permitan a los clientes acceder a la aplicación y poder consultar las dietas que les marquen los profesionales.

8 Bibliografía

- *Figma: the collaborative interface design tool*. (n.d.). Figma.
<https://www.figma.com/>
- *MySQL :: Developer Zone*. (s. f.). <https://dev.mysql.com/>
- *Node.js*. (s. f.). Node.js. <https://nodejs.org/en>

- *Express - Infraestructura de aplicaciones web Node.js.* (s. f.). <https://expressjs.com/es/>
- *auth0.com.* (s. f.). *JWT.IO.* JSON Web Tokens - jwt.io. <https://jwt.io/>
- *Vue.js - The Progressive JavaScript Framework* / *Vue.js.* (s. f.). <https://vuejs.org/>
- *Vue Router* / *The official Router for Vue.js.* (s. f.). <https://router.vuejs.org/>
- *Getting started* / *Vuelidate.* (s. f.). <https://vuelidate-next.netlify.app/>
-  *vue-chartjs.* (s. f.). <https://vue-chartjs.org/>
- *Chart.js.* (s. f.). Open source HTML5 Charts for your website. <https://www.chartjs.org/>
- *JavaScript With Syntax For Types.* (s. f.). <https://www.typescriptlang.org/>
- *Axios.* (s. f.). <https://axios-http.com/es/>
- *Pinia* / *The intuitive store for Vue.js.* (s. f.). <https://pinia.vuejs.org/>
- Contributors, M. O. J. T. A. B. (s. f.). *Bootstrap.* <https://getbootstrap.com/>
- *Sass: Syntactically Awesome Style Sheets.* (s. f.). <https://sass-lang.com/>
- *Docker: Accelerated, Containerized Application Development.* (2023). *Docker.* <https://www.docker.com/>
- *NGINX Product Documentation.* (s. f.). <https://docs.nginx.com/>
- *Git.* (s. f.). <https://git-scm.com/>