TFG: Entrega Final

Asignatura: Proyecto de desarrollo de aplicaciones web, 2º DAW

Autores: Jorge Jaén Ferrer; Raúl López López

Introducción

El presente documento tiene como finalidad exponer el trabajo realizado como proyecto de fin del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Web.

La información se presenta organizada en diferentes apartados que recogen la descripción de la aplicación web desarrollada, información relevante durante la toma de decisiones; las herramientas y lenguajes de programación empleados con dicho cometido; una descripción de cada una de las fases sucedidas hasta el resultado final, así como un apartado de anexos donde se recogen los recursos más relevantes utilizados durante todo el proceso.

Índice

1. Módulos formativos aplicados en el trabajo

Programación

Es una de las asignaturas que han supuesto un pilar fundamental para el desarrollo del proyecto, ya que precisamente ha brindado los conceptos básicos para saber escribir código

Entornos de Desarrollo

Esta asignatura brindó los conocimientos de Git básicos para estructurar y desarrollar un proyecto conjunto ya que el equipo estaba formado por más de una persona. En adición a esto se tuvo que aprender algunos comandos extras para perfeccionar el uso de la herramienta Git.

Lenguaje de marcas y sistemas de gestión de información

Esta asignatura ha aportado la base teórica y práctica necesaria para poder desarrollar y entender el desarrollo fundamentalmente con HTML, así como los fundamentos para comprender el ecosistema a nivel básico el entorno web.

Desarrollo de interfaces

Este módulo ha aportado al proyecto la organización y criterio necesarios para integrar los elementos gráficos de la manera más eficaz y eficiente posible de cara al usuario final, priorizando una experiencia de uso intuitiva.

Programación entorno cliente

Nuevamente, de este módulo se aplican conceptos necesarios para el desarrollo con lenguajes como JavaScript, así como la manipulación de los elementos que conforman la página web a través de funciones entre otras ideas.

Programación entorno servidor

Éste módulo ha supuesto un pilar importante en el desarrollo de aplicación ya que ha facilitando la comprensión y puesta en práctica de los elementos que conforman la lógica de negocio y base de datos de toda la aplicación entre otros conceptos.

Módulo MEAN Stack

Fundamentalmente ha aportado la herramienta fundamental empleada para el desarrollo del cliente: Angular; ha sido útil para organizar en componentes todos los elementos que conforman la UI, y “consumen” las REST api programadas.

Empresa e Iniciativa emprendedora

Sí bien no se ha utilizado de manera implícita en la programación de la aplicación, esta asignatura ha sido importante para la ideación del mismo: ha permitido aplicar ciertos criterios a la hora de diferenciar la funcionalidad de la app, que al fin y al cabo, está destinada a la operativa de un negocio.

2. Objetivos del proyecto

La razón de la elaboración de este proyecto viene dada por la necesidad que tienen los empleadores de una clínica de nutrición de estar informados en todo momento y en tiempo real de los datos propios de la actividad principal. Con la introducción de esta aplicación en la empresa se obtendrán múltiples beneficios tales como una mayor facilidad para el almacenamiento y gestión de los datos tanto del administrador como de los nutricionistas; una mayor posibilidad de dar a conocer sus servicios, más comodidad para los usuarios, etc.

3. Herramientas y lenguajes utilizados

La aplicación se desarrolla en su mayor parte usando lenguaje de servidor Java y Typescript en el cliente.

Cabe mencionar que todo el software utilizado es libre, con lo que la empresa tendrá un considerable ahorro en concepto de licencias. El software seleccionado no lo ha sido sólo por gratuito, sino porque además es unas de las tecnologías más utilizadas en la actualidad en el desarrollo de aplicaciones web debido a su fiabilidad y a su versatilidad, lo que se traduce en un gran soporte para posibles desarrollos posteriores, por su comunidad de desarrolladores que lo emplean, y las iteraciones que mejoran este lenguaje progresivamente.

Cabe añadir que la razón que más peso ha tenido a la hora de escoger este lenguaje, es la relación directa con el framework que se explicará más adelante en este apartado.

Motor de bases de datos
El motor de bases de datos usado es MySql. Las razones de su elección son las siguientes:

- Licencia de uso gratuita, lo que permite reducción de costes para el cliente.
- Es multiplataforma para Windows, Linux y Mac (los sistemas operativos más extendidos) con lo cual se podrá disponer de él en cualquiera de estos.
- Es un motor muy extendido en la comunidad de desarrolladores, con lo que conseguir soporte es muy sencillo.
- La labor de mantenimiento de una base de datos MySql es fácil debido a que presenta menos funciones frente a otros sistemas gestores. Sí bien puede parecer una desventaja al principio, tiene a su favor que el mantenimiento de la aplicación lo puede llevar el propio desarrollador, sin tener que recurrir a un administrador de bases de datos ampliando así el tiempo requerido para el desarrollo y en resolver incidencias
- Es escalable, lo cual nos da una ventaja con vistas al futuro.

Frameworks: Servidor y cliente.
El framework del lado del servidor usado es Spring 2.6.6. Las razones de su elección son las siguientes:

- Spring es completamente modular y soporta diferentes tecnologías como la inyección de dependencias, eventos, recursos, i18n, validación, enlace de datos, conversión de tipo, SpEL.
- Soporte DAO, JDBC, ORM, Marshalling XML.
- Gestión de transacciones.
- Comunicación remota, JMS, JCA, JMX, correo electrónico, tareas, programación, caché.
- Simulacro de objetos, el framework TestContext, Spring MVC prueba, WebTestClient.
- Permite la implementación de rutinas transversales.

- Facilita en gran medida la programación basada en MVC (Modelo Vista Controlador) y una implementación rápida basada en Inyección de Dependencias (Dependency Injection).
- Es un Framework que tiene un especial foco sobre la Seguridad.
- Compatible con Frameworks web: Spring WebFlux y Spring MVC.
- Permite el procesamiento de datos por lotes.
- A través de este módulo se puede configurar la visibilidad y gestión de los objetos Java para la configuración local o remota vía JMX.
- Es un framework liviano debido a su implementación POJO (Plain Old Java Object), Spring Framework no obliga al programador a heredar ninguna clase ni a implementar ninguna interfaz.

En cuanto al cliente, el servidor utilizado es Angular 13. Las razones de su elección son las siguientes:

- Facilita la organización y creación de proyectos gracias a su filosofía en propone organizar los proyectos en componentes.
- Usa lenguaje TypeScript. La documentación es más consistente porque la sintaxis y la forma de leer los códigos de la información es siempre la misma.
- Las aplicaciones son fáciles de mantener. Al usar TypeScript, cualquier cambio que deba hacerse en la aplicación podrá llevarse a cabo rápidamente minimizando errores.
- Hace posible la incorporación de programadores en los proyectos: cualquier programador de Angular puede leer el código escrito por otro programador de Angular y comprenderlo por la sintaxis sencilla.
  Esto es una gran ventaja a la hora de trabajar en equipo o de retomar proyectos inacabados.
- Utiliza componentes web: un componente web es una porción de código que puede reutilizarse en otros proyectos hechos con Angular. Además, son fáciles de convertir en componentes web nativos, por lo que pueden reutilizarse de nuevo en otro tipo de aplicaciones.
- El lenguaje garantiza estabilidad: con JavaScript los cambios son frecuentes, y puede resultar complicado aprender todas las novedades añadidas. Sin embargo, Angular es más estable en lo que se refiere a nuevas funcionalidades

Herramientas y lenguajes complementarios aplicados en el proyecto:

GitHub

Software de control de versiones empleado como repositorio de código donde centralizar todo el desarrollo del proyecto.

MySQL Workbech

Editor en el que se ha generado la base de datos del proyecto escogido por facilitar en gran medida el uso de SQL, y la modificación de las tablas del desarrollo,

Eclipse

Entorno de desarrollo concebido principalmente para programar con el lenguaje Java. Además, posee una gran integración con el framework Spring y todas sus "herramientas".

Postman

Software empleado para testear el funcionamiento de los métodos HTTP desarrollados en la parte de servidor.

VSCode

Entorno de desarrollo empleado para trabajar con los lenguajes y frameworks citados anteriormente para la parte de cliente.

Angular Material

Front End framework, el cual ha permitido desarrollar y maquetar con mayor velocidad la página web al dotar de estilos a la plataforma gracias a sus componentes

HTML/CSS

Lenguajes bases de etiquetado para el desarrollo básico de la página web.

Draw.io

Herramienta empleada para el diseño de los casos de uso de los usuarios de la aplicación

4. Componentes del equipo y aportación realizada por cada estudiante

La aplicación web ha sido desarrollada por Jorge Jaén Ferrer y Raúl López López. A nivel individual, Jorge Jaén Ferrer es graduado en relaciones públicas y marketing internacional por la Universidad de Alicante; y Raúl López López es graduado en Nutrición Humana y Dietética por la Universidad de Navarra. Actualmente, ambos son estudiantes del grado superior de diseño de aplicaciones web,

En base al bagaje académico comentado, que coincide con los intereses personales de cada alumno, la idea de desarrollar una herramienta para ayudar a los gestores de negocios de salud - concretamente a gestores de clínicas de nutrición - cobra sentido completamente.

En la tarea de desarrollar el flujo de tareas y funcionalidades que la herramienta facilita al gestor de la clínica, Raúl y Jorge han aportado de manera respectiva conocimientos sobre la manera o protocolo diario que lleva a cabo el nutricionista en consulta, así como ciertos conceptos de empresa para poder orientar.

Respecto a la programación de la aplicación, ambos integrantes del equipo - como se comentará más adelante en el apartado de fases del proyecto - han trabajado de manera equitativa tanto en el desarrollo de la parte del backend, como en la parte del cliente; esto incluye todas la tareas de las fases que se han seguido en el proyecto.

5. Fases del proyecto

El proyecto ha sido desarrollado esencialmente siguiendo las siguientes fases diferenciadas: ideación, planificación, desarrollo, documentación.

1. **Ideación:** Es la primera fase cuyo objetivo primordial ha sido establecer objetivos e interés en común para centrar la temática del proyecto.

2. **Planificación:** Una vez concebida la idea, el siguiente paso ha sido establecer unos criterios o guías para determinar los siguientes pasos a tomar. En términos prácticos, el primer desarrollo a entregar iba a ser la parte del servidor, y para la entrega final, la parte del cliente.

3. **Desarrollo:** Desde la autonomía y comunicación constante, los desarrollos iban sucediéndose de manera equitativa en torno a los servicios REST citados en apartados anteriores, siendo ésta la filosofía de trabajo aplicada para la primera y la segunda entrega.

4. **Documentación**: La última fase se ha dedicado a realizar la presente memoria que recoge información sobre el proyecto, así como la documentación del código de la aplicación.

5.1 Arquitectura de la aplicación

A continuación se describe la estructura de la aplicación en la parte del servidor y cliente

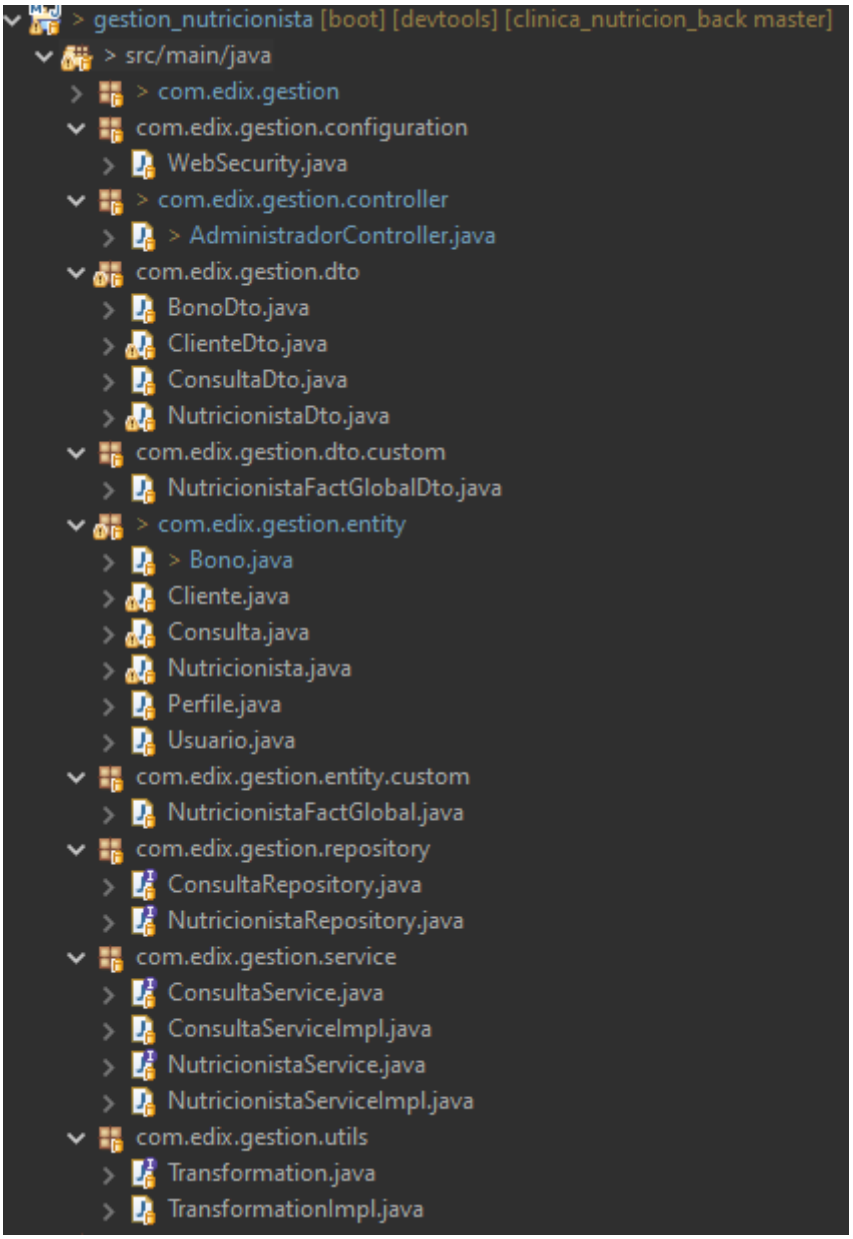

Imagen 3. Estructura parte servidor - Eclipse

El proyecto se elabora con una modelo-vista-controlador (MVC): esta arquitectura se caracteriza por separar los datos de una aplicación (modelo), la interfaz de usuario (vista), y la lógica de control (controlador) en 3 componentes distintos.

- El paquete *"configuration"* contiene el fichero donde se ubica la configuración de Spring Security

El resto de la estructura lo conforman los siguientes paquetes:

- **Entity**: Agrupan las entidades - o Java Beans - del modelos de datos
- **DTO** (Data Transfer Object): están basadas en las entidades, y poseen los atributos de éstas según conveniencia*****

  *****En este caso, se utilizaron para evitar datos anidados en las consultas con Postman
- **Custom entity**. Clase generada para realizar métodos HTTP relacionados con la facturación
- **Controllers**: En este paquete se ubican la lógica de las API (Application Programming Interface) que "consumirá" el cliente para manipular los datos
- **Service**: Incorpora la programación de la lógica de negocio, es decir, comprende los métodos que configuran cómo se manipulan los datos en la parte del servidor según el tipo de peticiones/tareas necesarias de la app
- **Repository**: Interfaz que acoge métodos propios de JPA, y configurados de manera específica para las consultas necesarias.

  En este caso, contiene querys para mostrar la facturación del negocio por nutricionistas
- **Utils**: Interfaz e implementación de la misma para programar la lógica entorno a los DTO

En cuanto a la estructura de la aplicación de Angular, los componentes se han organizado de la manera en la que se visualiza en la imagen inferior:

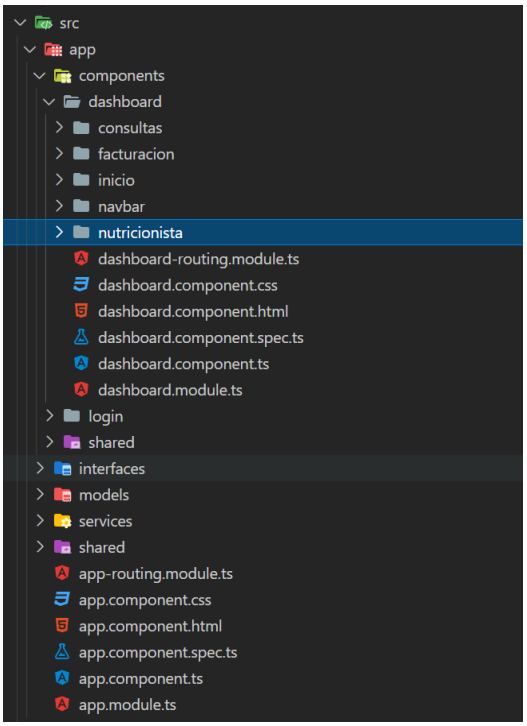

Imagen 3. Estructura parte cliente - Virtual Studio Code

Antes de pasar a explicar con más detalles la estructura de la aplicación, es preciso recalcar 2 conceptos básicos bajo los que Angular opera:

- Un componente está integrado por un esquema de 4 archivos: TypeScript, HTML, CSS y configuración de TypeScript
- Cada componente es potencialmente reutilizable dentro de la arquitectura de la aplicación, siendo cada uno de estos importado por cada componente que lo reutiliza

En base a lo anterior, podremos comprender el siguiente esquema de componentes:

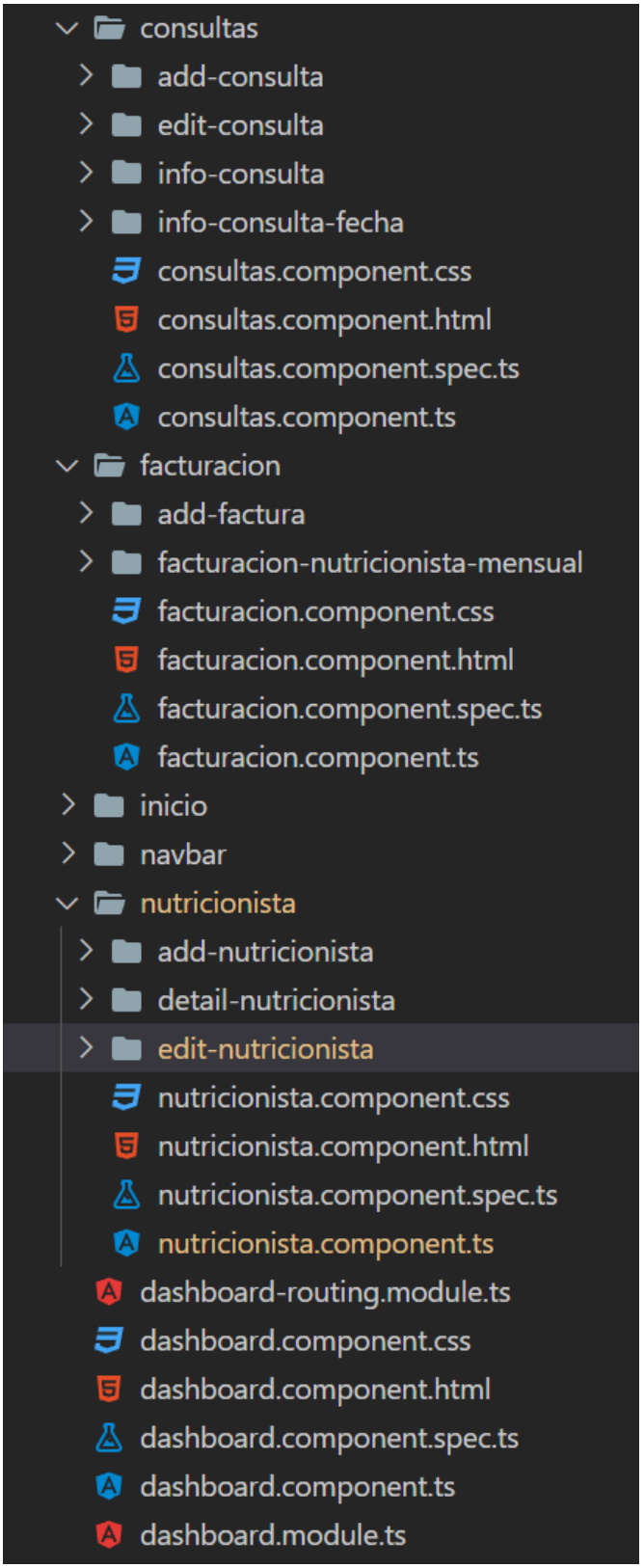

Imagen 4. Detalle de la estructura de componentes del proyecto

Los componentes principales son: ***dashboard*** **y** ***login***, incluyendo los archivos del esquema mencionado anteriormente. Dentro del primer componente se ubican con sus respectivos subcomponentes a cada tarea:

- ***navbar***
- ***consultas***
    - *add-consulta*
    - *edit-consulta*
    - *info-consulta-fecha*
- ***facturación***
    - *facturacion-nutricionista-mensual*
- ***nutricionistas***
    - *add-nutricionista*
    - *detail-nutricionista*
    - *edit-nutricionista*
    -
- ***inicio***

5.2. Modelo de dominio

El modelo de dominio representa la relación de las principales entidades que conforman la aplicación, traducido al concepto de programación orientada a objetos: aquellos objetos o representaciones interrelacionadas con la lógica de negocio del sistema desarrollado.

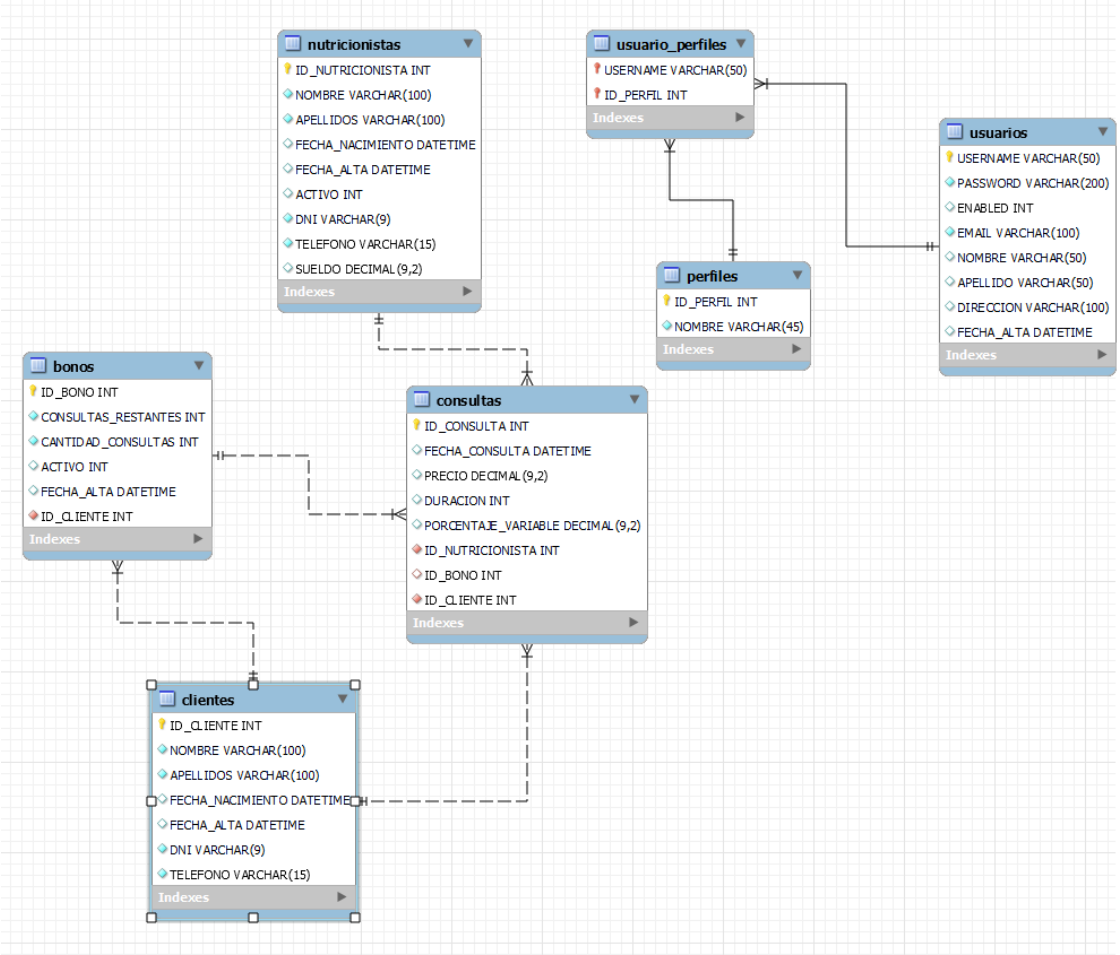

Imagen 2. Modelo de la base de datos

5.3. Casos de uso

Como su propio nombre indica, los casos de uso describen las funcionalidades a disposición para el usuario que utiliza la aplicación. En este caso, se conciben 2 tipos de usuario: administrador de la clínica, y usuario/nutricionista. A continuación se muestran en la siguiente imagen los principales casos de uso:

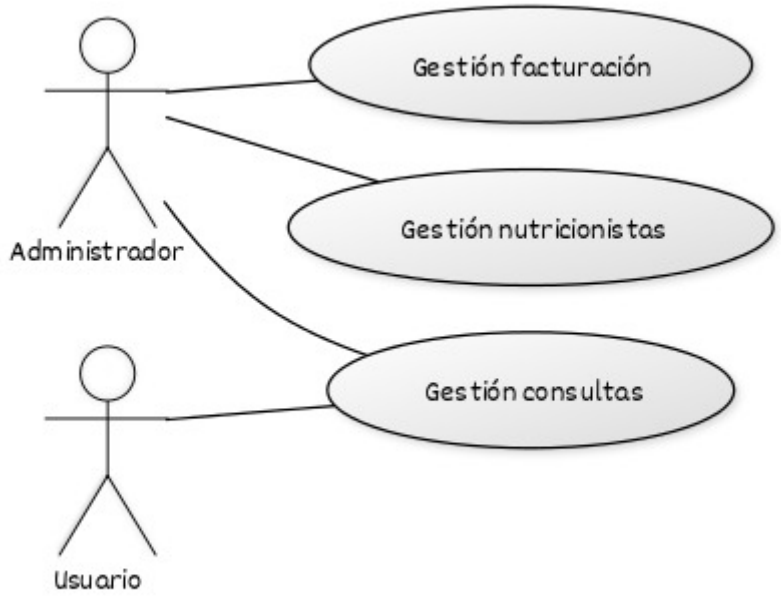

Imagen 1. Diagrama de casos de uso

5.4. Servicios REST de la aplicación

Para la entrega final del proyecto se ha decidido agrupar todos los servicios bajo un mismo controlador - cuyo @RequestMapping principal es **"/gestion"** - resultado en el siguiente esquema de métodos:

- Bajo el mapping del controlador funciona el servicio para mostrar todas las consultas
- **/consulta/{id}** → consulta por id
- **/nutricionista/consultas/{idNutricionista}** → todas las consultas por nutricionista
- **/nutricionista/consultas/{idNutricionista}/{fechaConsulta}** → todas las consultas por nutricionista para una fechaConsulta
- **/consultas/alta** → dar de alta una consultas
- **/consultas/actualizar** → actualizar una consulta
- **/consultas/eliminar/{idConsulta}** → eliminar una consulta
- **/nutricionista** → información de todos los nutricionistas
- /**nutricionista/{id}** → información de nutricionista
- **/facturacion** → facturacion global y por horas de cada nutricionista entre dos fechas
- **/facturacion/mensual** → facturacion global y por horas de cada nutricionista del mes actual
- **/facturacion/nutricionista/{id}** → facturacion global y por horas de un nutricionista del mes actual
- **/facturacion/nutricionista/{id}/mensual** → facturacion global y por horas de un nutricionista entre dos fechas
- **/facturacion/diaria** → facturacion global y por horas de cada nutricionista del día actual
- **/nutricionista/alta** → alta de un nutricionista
- **/nutricionista/actualizar** → actualizar un nutricionista
- **/nutricionista/eliminar/{id}** → eliminar un nutricionista

5.5. Instrucciones de despliegue

A continuación se indican los pasos necesarios para el despliegue:

1. **Parte servidor**: se instala el paquete *gestion_nutricionista* con el proyecto back de Spring y se ejecuta como aplicación spring boot app.
2. **Parte cliente**: hay que abrilo en VSCode o cualquier otro IDE y ejecutar el comando *npm install* para que cargue las carpetas necesarias. *Para desplegar se usa el comando **ng serve***.
3. **Base de datos**: tiene como nombre gestion_nutricionista y el script de creación es el siguiente adjuntado en anexos. gestion_nutricionista.sql

6. Conclusiones y mejoras del proyecto

La idea original es una aplicación que pueda ser utilizada en entorno real, por lo tanto, además de todas las consultas realizadas, se piensa que:

- Integración del login
- Integrar el usuario de nutricionista en el sistema
- Tareas de nutricionista:
    - Formulario Edit: recuperar los datos del nutricionista, para mostrarlos en sus respectivos campos
- Tareas de bono:
    - Mostrar los bonos asociados en la pantalla de nutricionista
- Tareas de cliente
    - Mostrar todos los clientes
    - Mostrar detalles de los clientes
- Tareas de facturación
    - Estilos: En la pantalla de inicio, mostrar los datos de manera más concisa empleando para ello gráficos, por ejemplo, los creados por Google.
    - Información sobre los datos: explicar los negativos, o en su defecto, mostrar sólo los datos cuando la facturación sea positiva (a partir de que el nutricionista haya cubierto los gastos en concepto de su salario)
- Mejorar de manera iterativa de la mano del cliente la interfaz de usuario: transformarla en PWA entre otras características