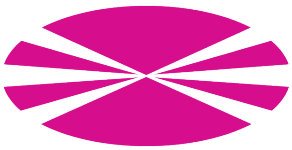

UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

TRABALLO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Aplicación web para a xestión de menús domésticos con servizos nutricionais : Eat Fit Week!

Autor: ***Elías Ferreiro Borreiros***

Director: ***Juan José Sánchez Penas***

A Coruña, Agosto, 2019

RESUMEN

Hoy en día, con el cambio en los estilos de vida de las personas y tendiendo hacia unas costumbres más sedentarias, hay una mayor necesidad de enfocarse en una dieta equilibrada y saludable. Para ello, se han desarrollado muchos sistemas webs y móviles para la gestión de comidas y de sus valores nutricionales. Sin embargo, analizando esos sistemas, vemos que tienen un error en su planteamiento al inundar a los usuarios con formularios sobrecargados y repletos de información innecesaria. El otro problema principal de estos sistemas es la cantidad exagerada de trabajo manual que debe hacer el usuario antes de poder disfrutar de la funcionalidad principal.

Para resolver todo esto, hemos decidido plantear el desarrollo de una aplicación que solvente estos problemas y ofrezca una funcionalidad que no disponen los competidores : el análisis nutricional dinámico de las comidas planificadas para la semana configurable por el usuario. está sobrepasando.

A mayores permitiremos la gestión de las entidades necesarias para esta planificación: ingredientes, platos, menús ... Esto se hará siguiendo la filosofía inicial del proyecto: simplificar la entrada lo más posible y disminuir el esfuerzo requerido por el usuario. Para esto llamaremos a servicios externos que nos permitirán estimar las características nutricionales de los ingredientes de forma que el usuario no tendrá que indicar esos datos y permitiremos con cada registro de usuario el alta automática de unos ingredientes base utilizables en la mayoría de recetas que agilizarán la configuración necesaria de un nuevo perfil para permitir disfrutar al máximo al usuario de las funcionalidades realmente importantes desde el momento más temprano posible.

Título: Aplicación web para a xestión de menús domésticos con servizos nutricionais

Autor: Elías Ferreiro Borreiros

Tutor/Director: Juan José Sánchez Penas

Palabras clave: Java EE, POJO, Maven, Angular JS, Spring, Hibernate, Web, MySQL, Tarea, Lista, Contexto, Cliente - Servidor, Food, Planning, Management, Scrum.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

IMPLEMENTACIÓN

1.1. Software requerido

Como se ha especificado con anterioridad, para la implementación del sistema hemos decidido el uso del IDE Eclipse, Spring, Hibernate y Spring Data para el acceso a datos y el modelo, Angular JS y Typescript para la interfaz visual y MySQL como base de datos.

1.2. Proceso seguido

Durante la implementación del **backend** se ha buscado en todo momento la **eficiencia**, minimizando el número de **consultas** necesarias y el número de **bucles** utilizados para el procesamiento de la información requerida.
En este segundo apartado cabe destacar que nos hemos aprovechado de las ventajas que ofrece Java 8 con sus **streams y lambdas** soportados para hacer toda posible iteración lo más eficiente posible y legible en el código resultante.
Nuestras elecciones tecnológicas con Spring Data como gestor de consultas y Mapstruct como mapeador de entidades de hibernate a dtos enviados en la respuesta REST nos han permitido garantizar un buen nivel de agilidad en el desarrollo del backend ante cambios sobre operaciones ya existentes que requieran un nuevo campo de respuesta al solo necesitar declarar el nuevo campo en la entidad de hibernate y

en el dto de respuesta haciendo Mapstruct el mapeo automático al utilizar el mismo nombre en los dos objetos.

Durante la implementación del **frontend** se ha buscado que los componentes definidos sean lo más acotados, independientes y reutilizables posibles para poder minimizar la repetición de código y aumentar su legibilidad. Se ha intentado que cada componente delege todos sus cálculos y lógica necesarios en sus servicio asignado y deje al controlador typescript las funciones de visualización de sus elementos HTML lo que hace que cada fichero tenga una responsabilidad clara y minimiza su número de líneas para seguir mejorando la legibilidad.

Nos hemos preocupado de definir los estilos comunes del sistema en el fichero CSS de nivel más raíz posible para evitar su repetición en cada una de las hojas de estilos de los componentes utilizados.

1.3. Algoritmo de generación de menú aleatorio

Dentro de la implementación del api rest queremos destacar el algoritmo de generación de menú aleatorio ya que lo vemos como un punto de interés en su implementación dentro de las diferentes operaciones rest que se aportan al frontal.

Dentro de MenuServiceImpl tenemos los siguientes métodos que se utilizan para obtener, a partir del id de menú, id de usuario y fechas de inicio y fin de generación del menú, un menú aleatorio entre las dos fechas.

```
@Override
@Transactional
public ResponseDto generateRandomMenu(Integer menuId, Integer userId,
    String startDate, String endDate) {
   // First, we query the necessary info : The menu, user dishes and
      number of meals
   Menu menu = this.repository.findOne(menuId);
   List<Dish> userDishes = this.dishRepository.findUserDishes(userId);
```

```
        Integer mealsInWeek = this.userService.findUserMeals(userId).size();

        ...

        // We iterate over the correct date range
        while (startDate.compareTo(endDate) <= 0) {
            // We fill each day with random valid dishes
            fillDayWithRandomDishes(menu, userDishes, startDate, endDate);
            startDate = startDate.plusDays(1L);
        }

        // We persist the changes on the menu
        menu = this.repository.save(menu);

        // We return a message indicating that the operation has been a
            success
        return new ResponseDto(ResponseDto.OK_CODE, "Menu random generated
            correctly");
    }

    private void fillDayWithRandomDishes(Menu menu, List<Dish> userDishes,
        LocalDateTime date, Integer mealsInWeek) {
        for (int i = 0; i < mealsInWeek; i++) {
            // Select a random valid dish
            Dish randomDish = selectRandomDishWithValidMeal(userDishes, date);
            if (randomDish != null) {
                // Add it to the menu
                MenuDisRel mdr = new MenuDisRel(new MenuDisRelId(menu,
                    randomDish,
                date.format(DateTimeFormatter.ofPattern("yyyy-MM-dd
                    HH:mm:ss.S"))));
```

```
            menu.getDishes().add(mdr);
        }
        date = date.plusHours(1L);
    }
}

// We'll try to select a dish whose meals are accord with the date
// If we can't, we return null
private Dish selectRandomDishWithValidMeal(List<Dish> userDishes,
    LocalDateTime date) {
    Dish validDish = null;
    String hourFormatted =
        date.format(DateTimeFormatter.ofPattern("HH"));
    // We filter the dishes by those with any available meal that
        matches the date
    List<Dish> validDishes = userDishes.stream().filter(dish ->
        dish.getMeals().stream()
    .map(meal -> meal.getId().getMeal()).anyMatch(m ->
        hourFormatted.equals(m.getHour())))
    .collect(Collectors.toList());
    if (!validDishes.isEmpty()) {
        validDish = randomSelectDish(validDishes);
    }
    return validDish;
}

private Dish randomSelectDish(List<Dish> userDishes) {
    Random random = new Random();
    Integer randomIndex = random.nextInt((userDishes.size() - 1) + 1);
    return userDishes.get(randomIndex);
}
```

En el método público que será llamado por el controlador indicamos a Spring que debe ser transaccional, de forma que en caso de haber cualquier error durante el mismo se debe hacer rollback de toda la operación para que no nos queden cambios a medias y un menú a "medio generar"

Dentro de este método iteramos por las fechas correctas para la generación del menú aleatorio, es decir, en caso de indicársenos un rango de fechas concreto, utilizaremos ese rango y, en caso de no indicársenos, utilizaremos toda la semana que representa el menú.

Por cada día dentro de ese rango, iremos seleccionando para sus comidas platos aleatorios únicamente dentro de los platos del usuario que tienen esa comida entre sus comidas permitidas (Para el "Desayunoütilizaremos los platos que estén permitidos en un "Desayuno"por ejemplo). Como comentábamos antes, utilizamos streams para hacer que esta filtración de la lista sea más eficiente.

Para mayor aleatoriedad, para cada selección aleatoria generamos un nuevo selector de números aleatorios para obtener el índice de la lista que corresponderá con el plato seleccionado.

Hemos definido cada método auxiliar debajo del método que lo requiere ya que esto mejora la eficiencia del compilador de Java al interpretarlos.