Universitat Autònoma de Barcelona

Máster en Tradumática: Tecnologías de la Traducción

Módulo I: Automatización de la traducción

Traducción de productos digitales

Profesor: Adrià Martín Mor

Proyecto de localización de VeganStats del inglés al español

Presentado por Leire Gar Bailo

Correo electrónico: leire.gar@e-campus.uab.cat

Grupo número 12

Barcelona, 9 de diciembre de 2020

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. Introducción

VeganStats es una app React Native programada en JavaScript que fue desarrollada por Mike Gibson, un estadounidense que trabaja como desarrollador de *software* autónomo. La app se lanzó en 2018 para dispositivos móviles Android y iOS y permite introducir la fecha en la que una persona comenzó su dieta vegana y, a partir de ahí calcular, el impacto positivo que esa decisión ha tenido en el mundo en cifras reales. Además, envía notificaciones con mensajes motivacionales cuando se alcanzan ciertos logros medioambientales o cuando se cumple cierto tiempo desde que se comenzó la dieta vegana.

Los principales motivos por los que se eligió realizar este proyecto sobre esta app fueron dos: en primer lugar, por compartir los principios ideológicos que se encuentran tras la creación de esta app y, en segundo, por un interés personal en trabajar en un proceso de localización que supusiera un esfuerzo en cuanto a la dificultad técnica y no tanto a la lingüística. Personalmente, contemplo la ingeniería de la localización como un campo en el que desarrollar mi carrera profesional y, por ese motivo, consideré este proyecto como una gran oportunidad para empezar a formarme.

Antes de comenzar con el proceso de localización, se estableció contacto con el desarrollador de la app para solicitar su permiso y, una vez obtenido el visto bueno, se inició la preparación del proyecto.

Nombre del producto	VeganStats
Número de palabras	1364
Combinación lingüística	EN > ES
Herramienta TAO	OmegaT
Otras herramientas	Notepad++, Rainbow, WinMerge

Tabla 1. Tabla resumen del proyecto de localización de VeganStats

En un principio, el número de palabras estimado fue de unas 2500. Sin embargo, tras el proceso de extracción de texto, se comprobó que el número exacto era de 1364 y que se había producido un error en la herramienta que se había utilizado para calcular las palabras, ya que no se habían configurado los filtros necesarios. Este volumen de palabras puede parecer insuficiente para un proyecto de esta envergadura, pero como se verá más adelante, la mayor parte de la carga de trabajo ha recaído sobre otras fases distintas a la de la traducción.

2. Descripción del proceso de localización

Según indican Martín Mor *et al.* (2016: 18), el proceso de traducción (o en este caso de localización) se divide en seis fases: la obtención, el análisis, la preparación, la traducción inicial, la postproducción y, finalmente, la entrega.

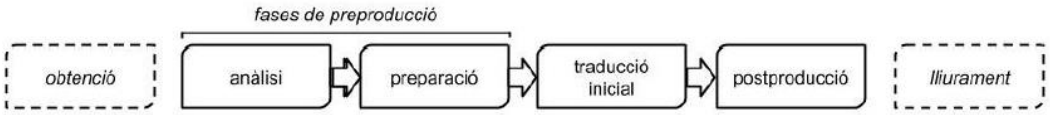


Figura 1. Fases del proceso de traducción según Martín Mor et al. (2016:18)

En los siguientes apartados se explicarán en detalle cada una de las tareas llevadas a cabo en cada una de estas fases, comenzando por la elección del programa y la obtención de los archivos y finalizando con la entrega de los archivos localizados.

2.1. Obtención

La obtención de los archivos que conforman la app VeganStats fue sencilla puesto que, como la mayor parte del *software* de código abierto, estos se encuentran disponibles para su descarga en GitHub.

2.2. Análisis

Una vez descargados los archivos del proyecto en el equipo de trabajo, se procedió a realizar un análisis de estos. En una primera instancia se observó que no todos los archivos debían traducirse. Por ejemplo, la carpeta contenía archivos en formatos no traducibles, como pueden ser las imágenes de los iconos, las fuentes de texto, los archivos para ejecutar la app, etc.

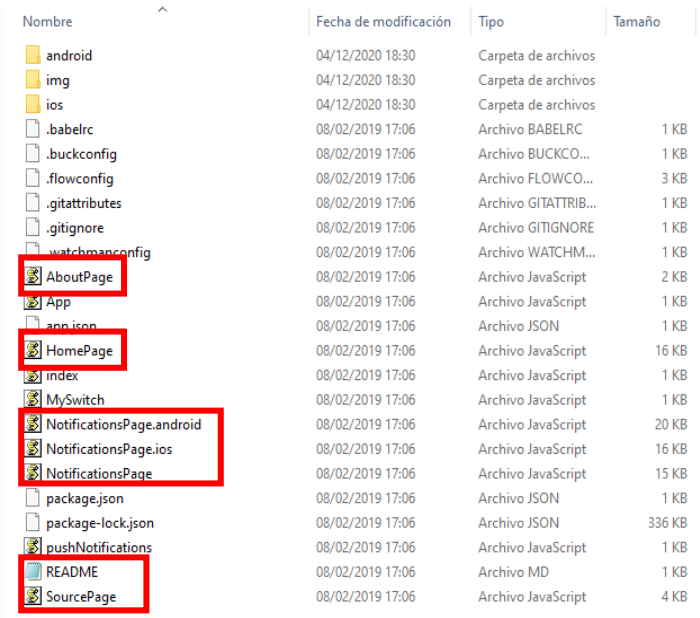


Figura 2. Archivos para localizar

Sin embargo, para poder comprobar qué archivos contenían cadenas de texto traducible y cuáles eran simplemente necesarios para el buen funcionamiento de la app, hubo que hacer un trabajo manual de comprobación. Para realizar

esta tarea se utilizó Notepad++. Tras comprobar todos los archivos uno por uno, se concluyó que el número de archivos que debían ser localizados eran siete: *HomePage*, *AboutPage*, *NotificationsPage.android*, *NotificationsPage.ios*, *NotificationsPage*, *README* y, finalmente, *SourcePage*.

Tras este análisis inicial de los archivos, se procedió a realizar un análisis de los formatos. La mayoría de los archivos (seis de siete) estaban en formato JavaScript (abreviado como JS), que es un lenguaje de programación interpretado que se utiliza principalmente para aplicaciones de escritorio (JavaScript, 2020), aunque en este caso se usó para desarrollar una app para dispositivos móviles.

Con la ayuda de Notepad++, se realizó un análisis de la estructura de estos archivos y se pudo observar que las cadenas de texto traducible se encontraban insertadas dentro del código de la app.

Figura 3. Ejemplo del código del archivo HomePage con el texto traducible insertado dentro del código

Esto quiere decir que, durante el proceso de diseño del producto, no se tuvo en cuenta la fase de la internacionalización, probablemente porque el desarrollador en ese momento no tenía la intención de que su app estuviera disponible en otros idiomas que no fueran el inglés. Esta falta de internacionalización hizo necesario llevar a cabo un proceso de ingeniería de localización que permitiera separar las cadenas traducibles del código.

2.3. Preparación

Tras analizar las diferentes posibilidades de *software* libre que existen para la extracción de cadenas de texto, se decidió optar por Rainbow, una herramienta de Okapi Framework que permite realizar diversas tareas relacionadas con la localización, entre ellas la extracción de texto. La principal razón por la que se eligió trabajar con esta herramienta fue que permite exportar directamente los paquetes de traducción

al formato de proyecto de OmegaT, que es la herramienta de traducción asistida por ordenador (TAO) con la que se había decidido trabajar desde un principio.

El proceso de extracción de texto resultó relativamente sencillo y se dividió en tres partes: la importación de los archivos originales a Rainbow, la creación de las expresiones regulares (en adelante «RegEx») que permitirían extraer el texto y, finalmente, la creación del kit de traducción.

La parte más complicada del proceso fue, sin ninguna duda, la configuración de las RegEx. Gracias a la función de configurar filtros que ofrece Rainbow, es posible crear filtros personalizados según las necesidades de los diferentes proyectos. En este caso, se utilizó un filtro de RegEx que permite crear expresiones para extraer determinados fragmentos de texto de un archivo dado.

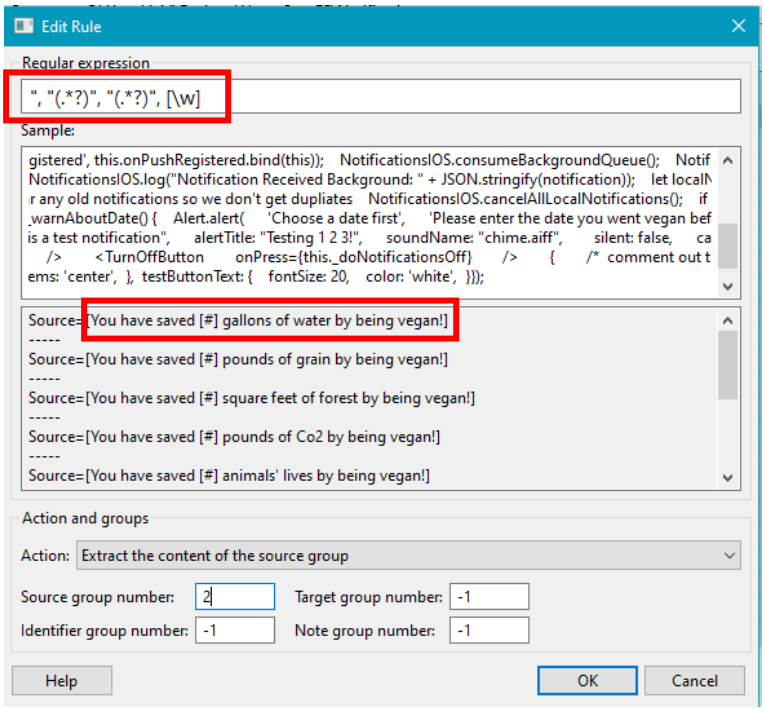


Figura 4. Ejemplo de RegEx para la extracción de texto en Rainbow

Como ya se ha dicho, esta fase supuso problemas y fue probablemente la parte más costosa de todo el proceso de localización. Sin embargo, la dificultad no se debía tanto a la estructura de las RegEx (la mayoría eran cortas y sencillas), sino a lo laborioso del proceso (en algunos de los archivos fue necesario crear hasta doce RegEx para poder extraer todas las cadenas traducibles).

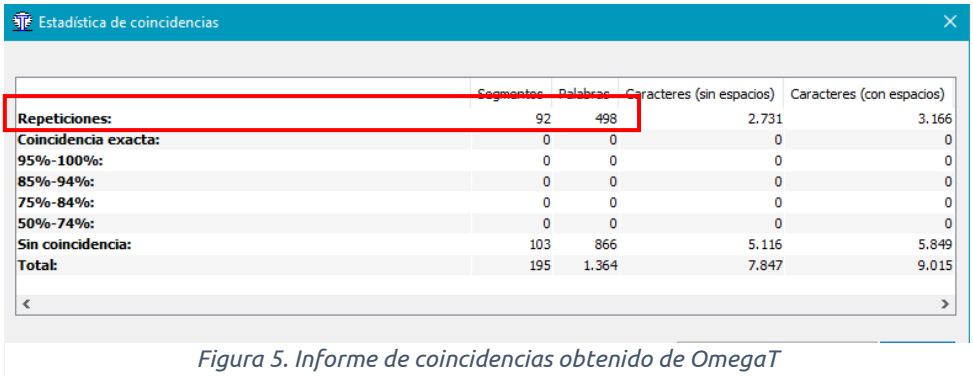
Una vez configuradas todas las RegEx, el tercer y último paso fue crear el kit de traducción que permite exportar el archivo únicamente con las cadenas traducibles para así poder llevar a cabo la fase de traducción en la herramienta TAO elegida. Rainbow cuenta con una función llamada «Kit Translation Creation» que permite la

exportación a varios formatos (XLIFF, paquete PO, RTF, proyecto de Transifex, proyecto de OmegaT, etc.). En este caso se decidió exportar los archivos como proyectos de OmegaT, ya que era la herramienta con la que se iba a realizar la traducción de los archivos y así se evitaba tener que crear los proyectos de manera manual.

Con la extracción de texto se dio por finalizado el proceso de preparación. Los archivos estaban listos para trabajar; sin embargo, antes había que realizar un análisis lingüístico del texto. Este análisis normalmente se realiza en la fase anterior, junto al análisis de datos y de formatos. Sin embargo, las características técnicas de este producto no habían permitido seguir el orden lógico del proceso de traducción.

En este análisis lingüístico se analizaron aspectos como la terminología, el estilo y, además, se identificaron los posibles problemas de traducción. En este momento del proceso también se decidió tomar como referencia principal la guía de estilo para la localización del inglés al español de Microsoft, debido a la falta de guías de estilo propias para la localización de apps para dispositivos móviles.

Por último y para finalizar ya el proceso de preparación, se calcularon las estadísticas del proyecto, para así tener una visión general de los archivos que se debían localizar. Al no utilizar ninguna memoria de traducción para el proyecto, el informe no aportó demasiada información. Sin embargo, en la Figura 5 se puede observar que casi un 37 % de las palabras se repiten en los diferentes archivos del proyecto, lo que redujo el tiempo de trabajo que se había planeado para la traducción en la siguiente fase.



	Segmentos	Palabras	Caracteres (sin espacios)	Caracteres (con espacios)
Repeticiones:	92	498	2.731	3.166
Coincidencia exacta:	0	0	0	0
95%-100%:	0	0	0	0
85%-94%:	0	0	0	0
75%-84%:	0	0	0	0
50%-74%:	0	0	0	0
Sin coincidencia:	103	866	5.116	5.849
Total:	195	1.364	7.847	9.015

Figura 5. Informe de coincidencias obtenido de OmegaT

2.4. Traducción inicial

Como ya se ha comentado anteriormente, la herramienta TAO escogida para llevar a cabo la fase de traducción de este proyecto fue OmegaT. Aunque *a priori* el texto parecía sencillo, a lo largo del proceso de traducción aparecieron numerosas dificultades, tal y como se verá más adelante en el punto *Análisis de la traducción*.

Además, al no disponer de memorias, la traducción se hizo directamente desde cero y, por tanto, no hubo preproducción.

2.5. Postproducción

Para explicar cómo se llevó a cabo el proceso de preproducción, se van a distinguir tres subfases o tareas: primero, la verificación lingüística; segundo, la preparación de los archivos localizados en formato original; finalmente, la verificación técnica.

2.5.1. Verificación lingüística

La verificación lingüística se realizó también con OmegaT. Para ello se utilizó, en primer lugar, la opción «Comprobar errores» de la herramienta. Esta función resultó ser muy útil, ya que permitió corregir errores tanto de terminología como de etiquetas y, además, gracias a la incorporación de la herramienta LanguageTool a OmegaT, también se pudieron detectar errores gramaticales y ortográficos.

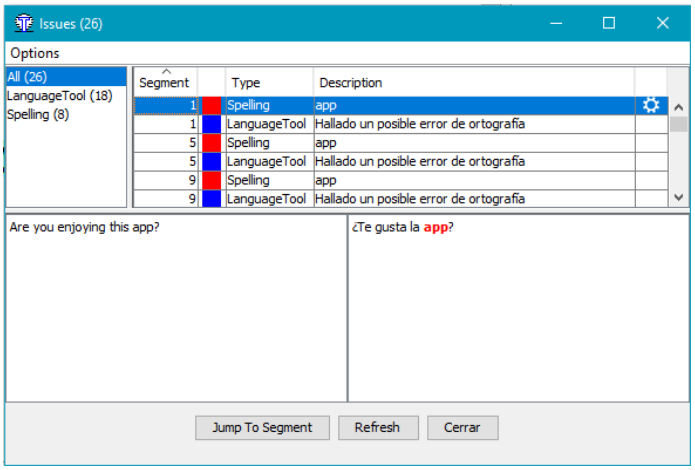


Figura 6. Función «Comprobación de errores» de OmegaT

Sin embargo, esta función de verificación lingüística que ofrece OmegaT es insuficiente para un proyecto con unas características técnicas tan complejas, ya que no permite, entre otros, detectar errores relativos a la puntuación o a la longitud de los segmentos. Por eso, para solucionar este problema, se decidió utilizar una de las *scripts* que vienen incluidas por defecto en carpeta de *Scripts* de la última versión de OmegaT y que se llama «QA-Check Issues». Con esta *script* fue posible identificar otra serie de errores como, por ejemplo, los dobles espacios, las diferencias de puntuación entre los segmentos originales y los meta, el uso de mayúsculas al inicio de un segmento, etc. (véase Figura 7). Sin embargo, para poder utilizar esta herramienta de verificación fue necesario configurar la *script* en el apartado de «Programación» de OmegaT y asignarle una ranura en la que ejecutarse, que en este caso fue la número siete.

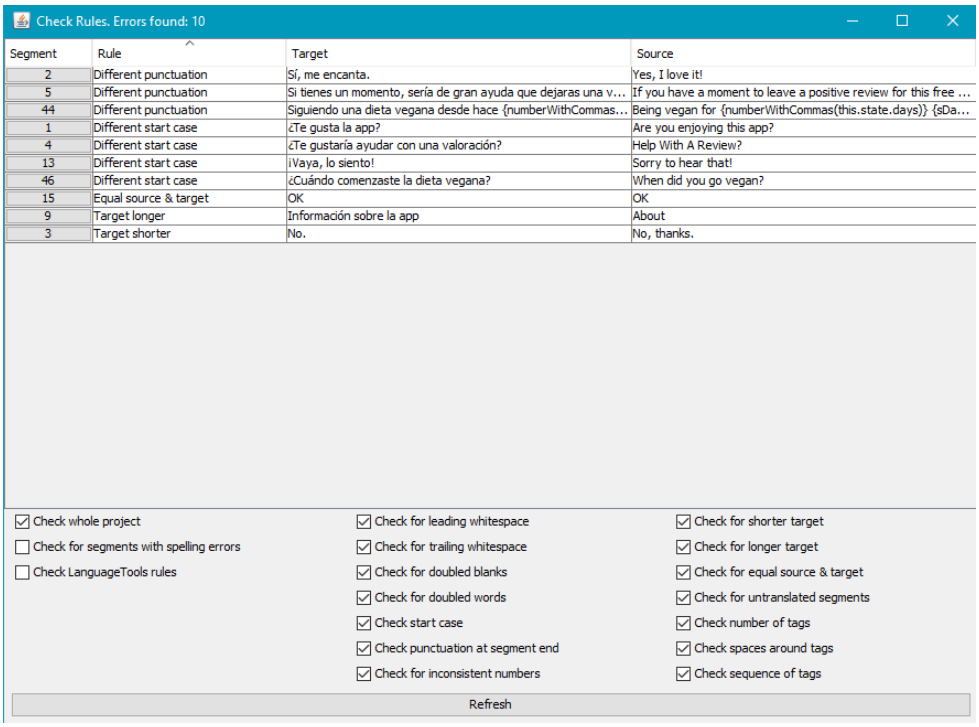


Figura 7. Script «QA-Check Issues» en OmegaT

Gracias a esta función se pudieron solucionar algunos de los errores cometidos durante la traducción inicial y que no se habían detectado con el comprobador de errores de OmegaT.

Por último, para finalizar el proceso de verificación lingüística, se llevó a cabo una revisión humana de todos los segmentos para comprobar, esta vez de manera manual, que no hubiera ningún error. A la hora de realizar la revisión se echó en falta una función de OmegaT que permitiera cambiar el estado de los segmentos para marcarlos como revisados. Gracias al trabajo de Marc Riera (Riera Irigoyen, 2018: 7), un compañero de máster de años anteriores, se encontró una solución para este problema: una *script* llamada «status_in_notes» que permite cambiar de manera muy sencilla el estado de los segmentos utilizando las notas. Por defecto, la *script* incluye tres marcadores de estado personalizables: <status:draft>, <status:needs-revision> y <status:approved> que fueron de gran ayuda durante la tarea de revisión.

Figura 8. Funcionamiento de la script «status_in_notes»

Una vez finalizada la revisión, se exportó a formato TMX una memoria de traducción con todos los segmentos del proyecto traducidos al español.

2.5.2. Preparación de archivos localizados

Tras terminar los procesos de localización y de revisión de los archivos que contenían las cadenas traducibles, llegó el momento de crear los archivos meta con el mismo formato que los archivos origen para así poder enviarlos para su publicación. Para ello se utilizó de nuevo la herramienta Rainbow, que cuenta con una función llamada «Translation Kit Post-Processing» que permite realizar este proceso.

Lo único que se necesita para poder crear este kit son el archivo manifest.rkm originado de manera automática al exportar las cadenas de texto traducible con la función «Translation Kit Creation», el archivo XLF traducido y el archivo original. Cabe destacar que es importante importar los archivos a Rainbow en este orden, ya que si no el programa nos muestra un error y no nos permite exporta correctamente los archivos localizados en formato original.

```
'use strict';

import React, {Component} from 'react';
import { Alert, AsyncStorage, Button, FlatList, Image, ScrollView, Sty
import { StackNavigator } from 'react-navigation';
import DatePicker from 'react-native-datepicker';
import Icon from 'react-native-vector-icons/MaterialCommunityIcons';
import RatingRequester from 'rn-rating-requester';
import SplashScreen from 'react-native-splash-screen';
import Switch from './MySwitch.js'

const RatingOptions = {
    enjoyingMessage: "¿Te gusta la app?",
    enjoyingActions: {
      accept: "Sí, ¡me encanta!",
      decline: "No.",
    },
    callbacks: {
      notEnjoyingApp: doNotEnjoyingApp,
    },
    title: "¿Quieres dejar una valoración?",
    message: "Si tienes un momento, sería de gran ayuda que dejaras un
    actionLabels: {
      decline: "Nunca",
      delay: "Más tarde",
      accept: "Valorar ahora"
    },
    eventsUntilPrompt: 3,
    daysBeforeReminding: 3,
    debug: false,
```

Figura 9. Archivos localizados en formato original

2.5.3. Verificación técnica

El proceso de verificación técnica se dividió en dos fases. Por un lado, una primera que permitiera comprobar que no se habían producido alteraciones en el código al extraer el texto original o al importar el texto localizado con la herramienta Rainbow. Por otro lado, la segunda de las fases consistiría en realizar un testeo, conocido en inglés como «end-to-end testing» (*E2E testing*), que sirviera para comprobar que el funcionamiento de la app fuera correcto de principio a fin y que la experiencia de usuario era buena.

Para poder realizar la primera de estas comprobaciones técnicas se utilizó WinMerge, una herramienta de código abierto que permite comparar dos archivos para observar las diferencias entre ambos. El uso de esta herramienta es muy sencillo, pues solo se deben importar los dos archivos que se desean comparar y WinMerge

muestra una especie de «radiografía» donde se observa, de una manera muy visual y sencilla de entender, las diferencias que hay entre los documentos.

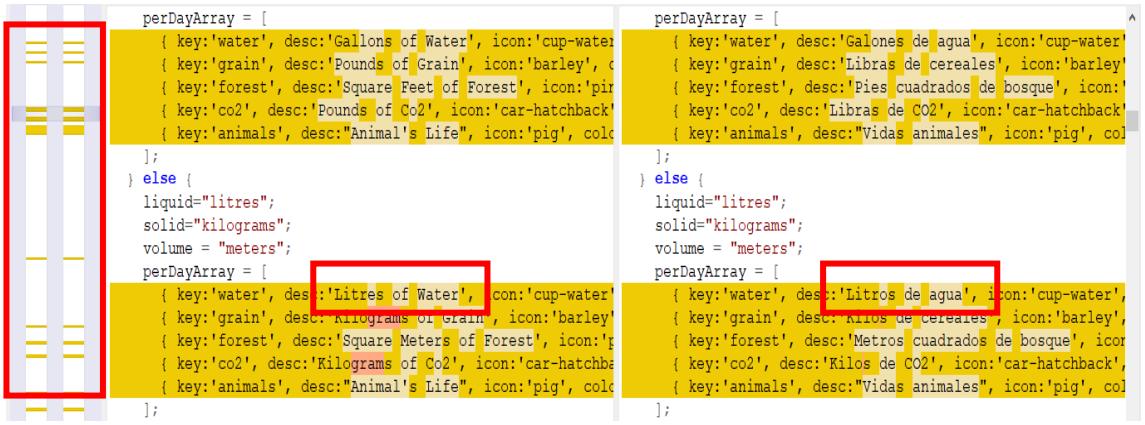


Figura 10. Diferencias entre dos archivos en WinMerge

Esta forma de verificación resultó de gran utilidad para este proyecto, ya que permitió comprobar que las únicas diferencias entre los archivos de origen y de llegada se encontraban en las partes donde aparecía texto traducible. Además, WinMerge también muestra la codificación y los saltos de línea de cada uno de los archivos, dos aspectos que suelen alterarse al exportar e importar archivos de unas aplicaciones a otras y a los que, por tanto, hay que prestar especial atención para así evitar problemas en el futuro.

Si bien la primera fase de la verificación técnica fue muy sencilla, el *E2E testing* produjo grandes problemas. La estructura de las apps React Native provoca que la fase de verificación técnica sea muy compleja y laboriosa, ya que es necesario programar los tests de tal manera que cualquier herramienta pueda acceder a las propiedades (Marcano, 2020).

Tras consultar varios manuales disponibles en Internet, se concluyó que la mejor forma de realizar el *E2E testing* era utilizar una herramienta de automatización de tests como, por ejemplo, Detox o Appium, ambas de código abierto. En sus respectivas ubicaciones web se pueden encontrar tutoriales muy detallados sobre cómo implementar estas herramientas para el testeo de apps React Native. Sin embargo, este proceso requiere de conocimientos avanzados sobre programación en JavaScript y, en mi caso, mis habilidades con este lenguaje son básicas. Por eso, tras varios intentos fallidos de llevar a cabo el *E2E testing* tanto con Detox como con Appium, se decidió contactar con el desarrollador de la app para pedirle ayuda, pero, desgraciadamente, no se obtuvo respuesta.

Así pues, el proceso de localización de esta app no se pudo completar de manera satisfactoria, puesto que la fase de testeo E2E es imprescindible para

comprobar si la app localizada cumple con las expectativas. La solución para este problema pasó por pedir al desarrollador en el momento de la entrega de los archivos localizados que, si le era posible, ejecutara los tests de verificación técnica y que comunicara cualquier error que pudiera encontrarse.

Este problema personalmente me provocó mucha frustración, ya que me hubiera gustado poder entregar un proyecto finalizado y no a medias. De todas maneras, esto puso de manifiesto un aspecto que nos han comentado en numerosas ocasiones a lo largo de estos primeros meses del máster y es que, en el mundo de la localización siempre existe la necesidad de contar con un equipo plural que pueda abarcar todo tipo de problemas.

2.6. Entrega

La última fase del proceso de localización fue la entrega de los archivos al cliente, que en este caso era el propio desarrollador de la aplicación, Mike Gibson. Los archivos localizados se enviaron vía correo electrónico.

A pesar de que el proyecto se entregó localizado, la creación de una app VeganStats en español es algo que requiere bastantes horas de trabajo de un desarrollador profesional y, por tanto, no creo que sea posible que esta vea la luz. Por ese motivo, se pidió al desarrollador que, si lo consideraba posible, publicara los documentos localizados al español en la carpeta de GitHub donde se encuentran alojados los archivos originales en inglés para que así estén disponibles para cualquier persona interesada en utilizarlos. Además, como ya se ha comentado anteriormente, también se solicitó al desarrollador que hiciera una verificación técnica E2E para detectar posibles problemas con el funcionamiento de la app. Desgraciadamente, no se recibió una respuesta.

3. Análisis de la traducción

Como se ha mencionado brevemente en el apartado relativo a la traducción inicial, durante el proceso surgieron numerosas dificultades de traducción, unas más fáciles de solucionar que otras.

Antes de pasar a explicar los ejemplos concretos que han supuesto problemas, cabe destacar un aspecto que se meditó mucho a la hora de traducir: el género de los nombres y de los adjetivos. En inglés, esto no supone ningún problema, ya que los nombres y los adjetivos carecen de flexión de género. Sin embargo, es un aspecto que se debe tener en cuenta a la hora de localizar al español.

Las posibles soluciones a este problema eran tres: utilizar el masculino genérico, desdoblar o neutralizar. Según un estudio (Allès *et al*, 2017), el 79 % de las personas veganas se identifican como mujeres. Por tanto, en una aplicación destinada a personas veganas, cabe asumir que la mayoría de los usuarios serán mujeres y, por eso, la opción de utilizar el masculino genérico quedó descartada. El desdoblamiento tampoco se consideró como una opción acertada para este proyecto, ya que resulta pesado y, por tanto, puede perjudicar la experiencia de usuario.

Así pues, la solución escogida fue neutralizar el género. Para ello se decidió sustituir el término «vegan» por «dieta vegana». Estos son algunos ejemplos:

ORIGINAL	TRADUCCIÓN
«When did you go vegan?»	«¿Cuándo comenzaste la dieta vegana?»
«Being vegan for X days.»	«Siguiendo una dieta vegana desde hace X días.»
«You have been vegan for X days/weeks/years.»	«Ya llevas X días/semanas/meses/años con tu dieta vegana.»

Tabla 2. Ejemplos de traducción del término «vegan»

La elección del término «dieta» en este contexto puede parecer arriesgada, ya que generalmente se utiliza como sinónimo de «régimen alimentario». Sin embargo, cuenta con una acepción en el *Diccionario de la lengua española* que la define como el «conjunto de sustancias que regularmente se ingieren como alimento». Además, el *Corpus del español* muestra que el uso de «dieta vegana» es bastante popular en la actualidad para referirse al tipo de alimentación que sigue una persona.

Una vez aclarado el aspecto del género, se procede a explicar con más detalle algunos ejemplos concretos que supusieron dificultades de traducción.

En algunos casos, fue suficiente con consultar la guía de estilo de Microsoft para comprobar cuál era la opción de traducción más adecuada. Algunos ejemplos de esto son el cambio de oraciones pasivas por oraciones activas o la traducción de las órdenes que se dan a los usuarios.

ORIGINAL	TRADUCCIÓN
«All future notifications from this app have been disabled»	«Se han desactivado las notificaciones para esta app»
«Clic here»	«Haz clic aquí»

Tabla 3. Ejemplos de traducciones sugeridas por la guía de estilo

Asimismo, siguiendo la guía de estilo, se decidió utilizar el término «app» en vez de «aplicación», como recomienda la Fundación del Español Urgente (2014).

En otras ocasiones, como por ejemplo en la traducción del título «About», se optó por no hacer caso a las recomendaciones de la guía de estilo, que sugerían traducirlo como «Sobre». Se consideró que esta opción de traducción no era correcta en español y, tras valorar y descartar varias alternativas, como por ejemplo «Sobre la app» (la información contenida en el apartado no es sobre la app, sino sobre el desarrollador) y «Sobre el desarrollador» (era demasiado largo para un botón de una app móvil), se decidió traducirlo como «Información», tal y como recomienda Muñoz Sánchez (2018).

También se encontraron problemas relacionados con el *locale*, por ejemplo en el caso de las unidades de medida o de los formatos de las fechas (este último causado por el aspecto técnico del proyecto y no el lingüístico, como se verá en el siguiente apartado). En cuanto a las unidades de medida, se decidió adaptar las unidades del sistema estadounidense a las del sistema internacional en los archivos *SourcePage* y *README*. Así pues, fue necesario recalcular las cantidades.

ORIGINAL	TRADUCCIÓN
«Each day, a person who eats a vegan diet saves 1,100 gallons of water, 45 pounds of grain, 30 sq ft of forested land, 20 lbs CO2 equivalent, and one animal's life.»	«Una persona que sigue una dieta vegana ahorra al día unos 416 litros de agua, unos 20 kilos de cereales y el equivalente a unos 4 kilos de CO2. Además, salva 2,7 metros cuadrados de superficie arbolada y la vida de un animal.»

Tabla 4. Ejemplo de problemas de «locale»

Otro aspecto que ha causado problemas es la tipografía. Debido a que en JavaScript programar el uso de letra cursiva es muy complicado, el desarrollador decidió utilizar comillas simples en vez de texto en cursiva. Así, en el caso de la traducción se hizo lo mismo, pero utilizando comillas angulares en lugar de simples. Un ejemplo de esto se puede ver cuando en la parte de la app referente a las fuentes se menciona el documental *Cowspiracy*. En el original el título del documental aparece entre comillas simples y, aunque en el archivo localizado se debería haber escrito en cursiva para así cumplir con lo que dicta la norma, el título se escribió entre comillas angulares debido a la dificultad de usar cursiva en JavaScript.

Para terminar con el análisis de la traducción, comentar un caso curioso. En el original, el verbo «to save» se utiliza para referirse tanto a litros de agua como a vidas animales. Sin embargo, en español no se puede utilizar el mismo verbo para ambos elementos, puesto que los litros de agua se ahorran y las vidas animales se salvan. En los casos en los que se trataba de un texto escrito en forma de oración, no hubo problema en utilizar más de un verbo para crear diferentes enunciados.

ORIGINAL	TRADUCCIÓN
«Each day, a person who eats a vegan diet saves 1,100 gallons of water, 45 pounds of grain, 30 sq ft of forested land, 20 lbs CO2 equivalent, and one animal's life.»	«Una persona que sigue una dieta vegana ahorra al día unos 416 litros de agua, unos 20 kilos de cereales y el equivalente a unos 4 kilos de CO2. Además, salva 2,7 metros cuadrados de superficie arbolada y la vida de un animal.»

Tabla 5. Ejemplo de traducción del verbo «to save»

Sin embargo, esto no fue tan sencillo en el caso de la pantalla de inicio (archivo *HomePage*). En la Figura 11 se puede observar que el título hace referencia a una serie de elementos y, por la forma en la que está dispuesta la información, no es posible utilizar más de un verbo, ya que resultaría confuso y perjudicaría a la experiencia de usuario. Dado que en español no existe un verbo polisémico con los significados de «ahorrar» y «salvar» (como sí ocurre en el caso del inglés con el verbo «to save»), fue necesario encontrar una solución alternativa. Así pues, se decidió traducir «Being vegan for X days saved:» como «Seguir una dieta vegana desde hace X días ha evitado la pérdida de lo siguiente:».

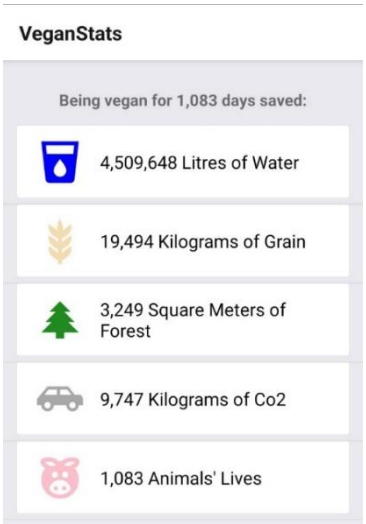


Figura 11. Página de inicio de VeganStats

4. Análisis técnico

Desde el punto de vista técnico, la parte más complicada del proceso de localización fue, como ya se visto en el apartado relativo a la preparación, la extracción del texto traducible. Sin embargo, el hecho de que los archivos estuvieran en un formato tan común como JavaScript facilitó el trabajo, puesto que no se dieron en ningún momento problemas de compatibilidad entre archivos y herramientas.

Asimismo, el hecho de trabajar únicamente con herramientas de código abierto (véase ANEXO I) y, en consecuencia, con archivos en formato estándar, ha facilitado el intercambio de datos de unas herramientas a otras.

No obstante, la complejidad técnica del proyecto ha causado que surgieran problemas. Por ejemplo, el ya mencionado anteriormente relativo a la imposibilidad de llevar a cabo el *E2E testing*. Por tanto, el desconocimiento sobre programación en JavaScript por parte de una localizadora novata como yo ha provocado que el resultado del proceso de localización no sea óptimo.

Dos aspectos de *locale* que no se han podido solucionar son la puntuación empleada en los números de más de cuatro cifras y el calendario que se emplea en la app para seleccionar la fecha en la que se inició la dieta vegana. Como ya se ha mencionado, la app se desarrolló teniendo en cuenta el *locale* estadounidense y, por tanto, está programada para que los números de más de cuatro cifras estén separados con comas y para que el calendario de la app, lógicamente, aparezca en inglés.

```
render() {
  const { navigate } = this.props.navigation;
  var sDays = "days";
  if (this.state.days === 1) {
    sDays = "day";
  }
  const header = this.state.calculated ?
    <Text style={styles.header}>Being vegan for {numberWithCommas(this.state.days)} {sDays} saved:</Text>
  //console.log("this.state:");
  //console.log(this.state);
  return (
```

Figura 12. Función que separa las unidades de millar con comas

La solución para estos dos aspectos pasa por hacer modificaciones en el código que permitan que, en vez de con comas, las unidades de millar se muestren separadas por espacios o puntos, como sería lo correcto en español. Sin embargo, para poder realizar estas modificaciones sin perjudicar la funcionalidad de la app, es necesario tener conocimientos de programación en JavaScript avanzados.

No obstante, hay otros aspectos que, aunque requieren de una intervención en el código, son más fáciles de solucionar. Por ejemplo, el botón «Metric« o «Unidades de

medida» viene desactivado por defecto y, por tanto, las unidades que se muestran al ejecutar la app soon las del sistema de unidades empleado en los Estados Unidos (véase Figura 11). Si en los archivos localizados se modifica el valor asociado a la variable «metric» del archivo *HomePage* de «false» a «true», entonces la app mostrará por defecto los valores con las unidades de medida de uso común en España. Sin embargo, no todos los países de habla hispana utilizan las mismas medidas, por eso es conveniente mantener la opción de utilizar otras unidades y no eliminar por completo la opción de cambiar de sistema de unidades.

```
if (metric == false) {
  perDayArray = [
    { key:'water', desc:'Gallons of Water', icon:'cup-water', color:'#00F', value:1100},
    { key:'grain', desc:'Pounds of Grain', icon:'barley', color:'#f5deb3',value:40},
    { key:'forest', desc:'Square Feet of Forest', icon:'pine-tree', color:'#228B22',value:30},
    { key:'co2', desc:'Pounds of Co2', icon:'car-hatchback', color:'#A9A9A9',value:20},
    { key:'animals', desc:"Animal's Life", icon:'pig', color:'#FFC0CB',value:1},
  ];

if (metric == true) {
  perDayArray = [
    { key:'water', desc:'Galones de agua', icon:'cup-water', color:'#00F', value:1100},
    { key:'grain', desc:'Libras de cereales', icon:'barley', color:'#f5deb3',value:40},
    { key:'forest', desc:'Pies cuadrados de bosque', icon:'pine-tree', color:'#228B22',value:30},
    { key:'co2', desc:'Libras de CO2', icon:'car-hatchback', color:'#A9A9A9',value:20},
    { key:'animals', desc:"Vidas animales", icon:'pig', color:'#FFC0CB',value:1},
  ];
```

Figura 13. Cambio del valor de la variable «metric»

Asimismo, es importante tener en cuenta que para que esto funcione también es necesario que cambiar los valores asociados a la variable «metric» en los archivos relativos a las notificaciones (*NotificationsPage*, *NotificationsPage.android* y *NotificationsPage.ios*). Así, si el botón de las unidades de medida está desactivado, las cantidades de los mensajes motivacionales que envía la app aparecerán en las unidades de medida de uso común en español de España, lo que garantiza una mejor experiencia de usuario.

5. Conclusiones

VeganStats parece, a primera vista, una app sencilla, tanto en el aspecto del diseño como en el de la traducción. Sin embargo, a lo largo del presente trabajo se ha podido comprobar que esto no es del todo cierto.

Desde el punto de vista de la traducción, se ha observado que algo tan común como un verbo («to save») puede provocar verdaderos quebraderos de cabeza. Además, también se ha puesto de manifiesto la importancia de tener en cuenta otros aspectos a la hora de localizar como, por ejemplo, el género. En una app como esta, donde los usuarios van a ser principalmente mujeres, no resultaría aceptable el uso del masculino genérico.

En cuanto al aspecto técnico, cabe señalar que hoy en día resultaría extraño recibir un encargo de localización tan complejo en cuanto al diseño y a la codificación como el que se presenta en este trabajo. Los desarrolladores, especialmente en las empresas globales, son conscientes de que internacionalizar es una forma sencilla de mejorar sus ventas y, en consecuencia, diseñan sus productos teniendo en cuenta el futuro proceso de localización. Sin embargo, es posible que haya desarrolladores pequeños, como es el caso de Mike Gibson, que decidan lanzar sus proyectos sin internacionalizar y que un día decidan que localizarlos les va a suponer un beneficio. Por eso es importante que, como localizadores, seamos capaces de enfrentarnos a todo tipo de encargos y situaciones.

Aun así, no hay que olvidarse de que el trabajo de un localizador es localizar, es decir, transferir información de una lengua a otra teniendo en cuenta una serie de aspectos culturales. Por mucho que un localizador profesional deba tener conocimientos en otros campos como la programación, puede que se encuentre con problemas que no sea capaz de resolver. Por eso es importante disponer de otra serie de aptitudes como puede ser la capacidad de contactar con otros profesionales que puedan ayudar a solucionar determinados problemas. En este caso concreto, sería necesario pedir ayuda a un desarrollador experto en JavaScript o React Native que ayudara a solucionar problemas como el de verificación técnica o el de las comas en los números de más de cuatro cifras.

Por último, desde una perspectiva más personal, me gustaría destacar que el resultado del proyecto no es del todo satisfactorio, principalmente debido a que no se han podido completar todas las fases del proceso de localización. Sin embargo, gracias a este trabajo se han podido aplicar a la práctica muchos de los conocimientos

adquiridos a lo largo de los primeros meses del máster, como puede ser la configuración de RegEX o el uso de Rainbow. El mayor reto probablemente haya sido llevar a cabo todo el proceso de localización utilizando únicamente herramientas de código abierto, algo que nunca me había planteado y que, sinceramente, ni siquiera consideraba posible. Gracias a este proyecto he descubierto que existe todo un mundo de herramientas gratuitas y de código abierto que prácticamente cumplen las mismas funciones que otras herramientas de pago. Asimismo, he aprendido a encontrar soluciones éticas y gratuitas a problemas que hace unos meses no me hubiera considerado capaz de solucionar y esto es algo que, sin ninguna duda, me será útil en mi futuro profesional.

6. Bibliografía

Allés, B., Baudry, J., Méjean, C., Touvier, M., Péneau, S., Hercberg, S. y Kesse-Guyot, E. (2017). Comparison of Sociodemographic and Nutritional Characteristics between Self-Reported Vegetarians, Vegans, and Meat-Eaters from the NutriNet-Santé Study. *Nutrients*, 9(9): 10-23. doi: 10.3390/nu9091023

Aplicación, alternativa a *app*. (2014, marzo 26). [Entrada de blog]. Fundéu. Recuperado de https://www.fundeu.es/recomendacion/aplicacion-alternativa-a-app/ (Consulta: 26/11/2020).

JavaScript. (2020, diciembre 7). *Wikipedia, La enciclopedia libre*. Recuperado de https://es.wikipedia.org/w/index.php?title=JavaScript&oldid=131517838 (Consulta: 25/11/2020).

Kos, I. (2018, febrero 12). #OmegaT Segments Status in Notes [Entradas de blog]. Translator's Recipes straight from the chef. Recuperado de https://libretraduko.wordpress.com/2018/02/12/omegat-segment-status-in-notes/ (Consulta: 30/11/2020).

Lipps, J. (2017, marzo 29*). An Introduction to Appium Desktop* [Vídeo]. Sauce Labs. Recuperado de https://www.youtube.com/watch?v=IOSUBda2-g4&t=1204s&ab_channel=SauceLab (Consulta: 1/12/2020).

Marcano, P. (2020, mayo 14). Setting up Appium for React Native e2e Testing [Entrada de blog]. Kaizen Softworks. Recuperado de https://blog.kzsoftworks.com/e2e-tests-appium-ios-android-react-native/ (Consulta: 2/12/2020).

Martín-Mor, A., Piqué, R. y Sánchez-Gijón, P. (2016). La digitalització del procés de traducció. En Biblioteca de traducció i interpretación (Ed.). *Tradumàtica. Tecnologies de la traducción* (pp. 22-35). Barcelona, España: Eumo.

Microsoft. (2019). *Spanish (Spain) Style Guide*. Recuperado de https://www.microsoft.com/en-us/language/styleguides (Consulta: 20/11/2020).

Mironov, R. (2013, diciembre 26). *Most Useful Scripts in OmegaT* [Vídeo]. Recuperado de https://www.youtube.com/watch?v=23pdUu4RebA&ab_channel=RomanMironov (Consulta: 30/11/2020).

Muñoz Sánchez, P. (2018, febrero 19). Cómo traducir la sección «About» en software y sitios web [Entrada de blog]. Algo más que traducir. Recuperado de https://algomasquetraducir.com/traducir-la-seccion-about-software-sitios-web/ (Consulta: 26/11/2020).

Rainbow Translation Kit Merging Step. (2020). [Entrada de blog]. Okapi Components. Recuperado de https://okapiframework.org/wiki/index.php/Rainbow_Translation_Kit_Merging_Step (Consulta: 28/11/2020).

Riera Irigoyen, M. (2018). *Projecte de localització del Duplicati al català* (Trabajo de Traducción de productos digitales). Universitat Autònoma de Barcelona, Cataluña. Recuperado de https://ddd.uab.cat/record/203604 (Consulta: 30/11/2020).

Text Merging Utility. (2020). [Entrada de blog]. Okapi Components. Recuperado de http://okapi.sourceforge.net/Release/Utilities/Help/merging.htm#Options_Options (Consulta: 28/11/2020).

Torres, T. (2020, junio). Test your React Native app with Detox + Q&A [Vídeo]. TestingAR. Recuperado de https://www.youtube.com/watch?v=FmIxCXy8BR0&t=846s&ab_channel=TestingAR (Consulta: 1/12/2020).

7. Anexos

Anexo I

Herramienta	Descripción	Parte del proceso de traducción en el que se ha utilizado
Notepad++	Editor gratuito de texto y de código fuente libre con soporte para varios lenguajes de programación.	Análisis
OmegaT	Herramienta TAO multiplataforma y gratuita que facilita las tareas de traducción gracias al uso de memorias de traducción.	Traducción y verificación lingüística
Rainbow	Herramienta multiplataforma y gratuita de Okapi Framework que sirve para facilitar varias tareas de localización.	Preparación de archivos y postproducción
WinMerge	Herramienta gratuita para Windows que permite comparar archivos y unirlos.	Verificación técnica

Tabla 6. Resumen de las herramientas utilizadas en el proceso de localización

8. Adenda

Tras presentar el proyecto de traducción de productos digitales el día 15 de diciembre de 2020, se recibió respuesta del desarrollador de la app, Mike Gibson, a los correos que se le habían enviado anteriormente. En su respuesta, el desarrollador pidió que se subieran los documentos localizados a la carpeta de GitHub donde se encuentran alojados los archivos originales de VeganStats. Asimismo, confirmó que él se encargaría de hacer el *E2E testing* que no se había podido realizar durante el proceso de localización y que intentaría, por todos los medios, hacer que la app VeganStats estuviera disponible en español en las diferentes tiendas de aplicaciones para dispositivos Android y iOS (con la respectiva mención a la localizadora).

Así pues, puede decirse que el proceso de localización se da por finalizado tras la realización de la fase de testeo por parte del desarrollador y, con suerte, el producto localizado verá la luz en los próximos meses.