

Ticket Translator

Simuñé Jaime Valencia, Erick Fernando Rivas Colunga

Published
with GitBook



Tabla de contenido

Introduction	0
Primeros Pasos	1
Obteniendo el codigo	1.1
Instalando Dependencias	1.2
Breve introducción a npm	1.2.1
Probando el programa	1.3
Tickets	2
Descripcion	2.1
Estructura	2.2
TicketTranslator	3
Uso	3.1
Mensajes	3.2
Base de Datos	4
SQLite	4.1
DDL	4.1.1
Ejemplos Consultas	4.1.2

Ticket translator

¿Que es esto?

TicketTranslator es un pequeño software que cumple con la finalidad de almacenar nuestros archivo de tipo `ticket` dentro de una base de datos.

Version

Actualmente TicketTranslator se encuentra en la version 1.1.0 esta nomenclatura indica que la version 1 ha recibido una actualizacion de mejoras.

¿Que incluye esta versión?

Esta versión cuenta con:

- Sistema de lectura de carpeta
- Lectura de archivos por extensión personal
- Almacenamientos con SQLite

Futuras versiones

- Selección de manejador con configuraciones en JSON
- Exclusión de archivos por carpeta

Contacto

Cualquier asunto sobre este software y su uso, puede ser tratado directamente con **Sinuhé Jaime Valencia** en el correo sierisimo@gmail.com o con **Erick Fernando Rivas Colunga** en el correo raycen.pain.skill@gmail.com

Primeros Pasos

En este capitulo se listan todas las partes necesarias para poder empezar a usar el software, esto involucra:

- Obtener el programa
- Obtener e instalar software
- Instalar las dependencias
- Probando el programa

Obteniendo el código

La manera mas simple de obtener el código es descargandolo directamente del repositorio publico en [Github](#) y bajando el archivo [.zip](#) que ahí se proporciona.

Esto descargara la ultima versión publicada.

Git

Si se desea mantener el proyecto al tanto, es posible utilizar la herramienta de manejo de código **GIT**.

Para obtener este proyecto, es necesario tener git instalado o una herramienta grafica que permita manejar proyectos git.

La manera mas rapida, desde una terminal es escribiendo:

```
git clone https://github.com/sierisimo/ticket-homerwork.git
```

Subversion

La herramienta subversion tambien esta soportada por el sitio de github, por lo cual es posible obtener el código escribiendo en una terminal:

```
svn co https://github.com/sierisimo/ticket-homerwork
```

Git + Subversion

Dada la capacidad de git para manejar proyectos de subversion es posible clonar el proyecto con git escribiendo en una terminal:

```
git svn clone -s https://github.com/sierisimo/ticket-homerwork
```

Instalando Dependencias

Node.js

[Node.js](#) es un programa que sirve como interprete de código JavaScript del lado del servidor, esto permite ejecutar aplicaciones escritas en JavaScript en cualquier computadora sin necesidad de un navegador web.

Es necesario tenerlo instalado para ejecutar el TicketTranslator.

Existen instaladores para casi cualquier distribución GNU/Linux, sistemas Windows o Mac OS X, incluso existen instaladores para sistemas BSD y en caso de no encontrar listado el instalador necesario se puede compilar el código fuente que también se encuentra disponible en el sitio oficial de Node.js

npm

npm es la utilidad que se incluye con Node.js y su nombre representa las siglas **node package manager**, es decir, es el manejador de paquetes de Node.js.

Su instalación es relativamente sencilla porque se incluye junto con la instalación de Node.js pero de no haberse instalado se puede obtener en:

<https://www.npmjs.com/package/npm>

Breve introducción a npm

¿Qué es?

npm es una herramienta como las que la se vienen presentando últimamente con muchos lenguajes de programación, un manejador de paquetes.

Se le puede comparar con herramientas como maven, gem, pip, easy_install, CPAN, etc. sin embargo su funcionamiento es mas sencillo, pues busca que toda la información de un proyecto se mantenga constante, también, esto involucra mantener sus dependencias con otros proyectos o paquetes configurables y de facil instalación.

package.json

La manera en que npm permite mantener un proyecto es mediante un archivo de tipo **JSON** en el cual se proporcione la información del proyecto según se quiera configurar para futuras referencias.

La manera de iniciar este JSON de forma sencilla es escribiendo en una terminal posicionada en la ruta de la raíz de nuestro proyecto:

```
npm init
```

Esto llevara por una guia rapida de como configurar un proyecto, desde definir quien es el autor, hasta el archivo principal por el cual se iniciara el proyecto.

Usando npm en un proyecto ageno

Cuando tenemos un proyecto que esta construido usando dependencias de npm, necesitaremos instalar esas dependencias antes de poder iniciar este proyecto, para hacerlo solo basta con colocar una terminal en la raíz del proyecto y escribir:

```
npm install
```

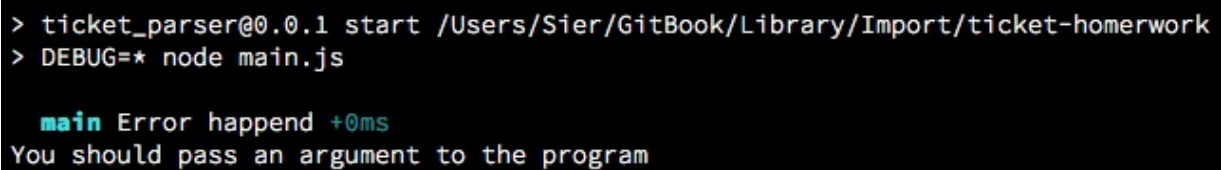
Esto indicara que deseamos buscar las dependencias listadas en el archivo `package.json` y que deseamos tenerlas localmente para poder continuar.

Probando el programa

Una vez realizados los pasos citados en los capítulos 1.1, 1.2 y 1.2.1 (es decir, instalado node.js, npm, clonado el proyecto e instalado dependencias con `npm install`) bastara con probar el programa, esto se hara posicionando una terminal en la carpeta del proyecto principal y escribiendo:

```
npm start
```

El programa tratara de iniciar y debera mostrar un error similar al siguiente:

A terminal window with a black background and white text. The first line shows the command prompt and the command being run: `> ticket_parser@0.0.1 start /Users/Sier/GitBook/Library/Import/ticket-homerwork`. The second line shows the command being executed: `> DEBUG=* node main.js`. The third line shows the output: `main Error happend +0ms`. The fourth line shows the error message: `You should pass an argument to the program`.

```
> ticket_parser@0.0.1 start /Users/Sier/GitBook/Library/Import/ticket-homerwork
> DEBUG=* node main.js

main Error happend +0ms
You should pass an argument to the program
```

Si este error aparece en nuestra pantalla, quiere decir que hemos instalado y probado TicketTranslator con exito.

Tickets

TicketTranslator puede ser interpretado en funcionalidad por su nombre, sin embargo si se quiere indagar un poco mas en que se hace, este capitulo lo explica mas detalladamente. Las siguientes 3 secciones explican el funcionamiento de los "tickets" y como es posible crear nuestros propios tickets.

Descripción

Un ticket, es en realidad un archivo en texto plano con una cierta estructura (revisar capítulo 2.2) que indica una transacción realizada en algún sitio.

Este archivo de texto, sirve como ejemplo de un ticket y permite a TicketTranslator demostrar el funcionamiento de un sistema de *parseo* (traducción) de datos.

La manera de notar estos archivos, es simplemente creando un archivo que tenga la extensión *ticket* y colocandolo en una carpeta con otros archivos con la misma extensión.

También es posible usar otras extensiones, para mas detalles sobre esto, consultar el artículo 3.4 y sus subartículos

Estructura

Los archivos de extensión ticket deben cumplir con una estructura mas o menos estricta, esta estructura se define a continuación.

Elementos

Cada archivo de ticket debe contener 3 tipos de elementos:

- Header/Descriptor
- Items
- Total/Resumen

Estos elementos se encuentran de alguna manera "ofuscados" para evitar que se pueda directamente ver el contenido sin antes pasarlo por una serie de programas.

Se asumen muchas cosas de estos elementos, sin embargo la estructura se mantiene.

Header/Descriptor

El header de cada ticket es una sola linea que debe contener los siguientes elementos en orden y tamaño estricto:

- **Indicador de linea:** denotado por una letra ***H***
- **Número de factura:** Una combinación de caracteres alfanuméricos con una longitud exacta de 8 elementos. Los primeros 3 elementos denotan el tipo de operación.
- **Número de cliente:** 6 caracteres alfanuméricos exactamente, estos sirven para identificar al cliente.
- **Fecha:** Fecha dada por el formato YYYYMMdd donde
 - Y: Dígito de año
 - M: Dígito de mes
 - d: Dígito de día
- **Tipo de moneda:** De dos a tres caracteres que denotan el tipo de moneda usada en la transacción.

Ejemplo

HINV03126R007PL20150317EUR

Items

Los items son el único elemento que puede estar una o mas veces repetido. Debe contener:

- **Indicador de linea:** denotado por una letra **I**
- **Id de producto:** Exactamente 7 elementos alfanuméricos.
- **Espacio divisor:** Un espacio para indicar separación entre elementos.
- **Antigüedad de producto:** Un número indicando la antigüedad del producto, no importa la cantidad de digitos en este número siempre que se respete el orden.
- **Espacio divisor:** Tres espacios para indicar separación
- **Cantidad:** Un número indicando la cantidad del producto, no importa la cantidad de digitos en este número siempre que se respete el orden.
- **Espacio divisor:** Cuatro espacios para indicar separación entre elementos
- **Valor neto:** Un número con punto decimal, sirve para indicar el precio.

Ejemplo

El siguiente ejemplo no usa espacios, usa el caracter ":" para proporcionar una mejor legibilidad del ejemplo

ICONU018:3:::5:::53.74

Total

La linea de total debe ser unica, sus elementos son evaluados y de no coincidir con lo necesario, seran reemplazado con los valores calculados.

Su estructura consiste:

- **Indicador de linea:** denotado por una letra **T**
- **Espacio divisor:** Un solo espacio para indicar separación.
- **Total de lineas:** Un número indicando el total de items, si este número no coincide, se usara el total de elementos en la parte de items
- **Espacio divisor:** 6 espacios para indicar separación entre elementos.
- **Total neto:** Un número con punto decimal que indica el total de la compra, si este número no tiene coherencia con la suma total de los items, se usara la suma total en lugar de este numero.

Ejemplo

El siguiente ejemplo no usa espacios, usa el caracter ":" para proporcionar una mejor legibilidad del ejemplo

T 5 1599.46

TicketTranslator

El proyecto esta construido enteramente en JavaScript y utiliza como manejador de base de datos SQLite en su versión 3.

Cuenta con un modulo que lee archivos, uno para hacer parseo de los datos, un modulo validador de los datos parseados y por ultimo un modulo que se encarga de almacenar en una base de datos local.

Cuenta con una serie de mensajes que indican que modulo ha encontrado errores o detalles que requieren atencion.

Uso

Para correr el programa basta con seguir la siguiente nomenclatura:

```
npm <action> <tickets_folder> [tickets_extension]
```

Donde los elementos encerrados entre `<` y `>` son obligatorios y deben existir.

<action>

Para esta version la unica acción disponible es: `start`

<tickets_folder>

Debera ser una carpeta dentro del sistema operativo validad, esta debe solo contener archivos con la extensión `ticket` a menos que se indique el parametro opcional `ticket_extension` en cuyo caso todos los archivos dentro de la carpeta deberan tener esa extensión.

Mensajes

Durante una ejecución normal se podran apreciar estos mensajes:

```
npm start && cd ..
> ticket_parser@0.0.1 start /Users/Sier/git/ticket-homerwork
> DEBUG=* node main.js "data"

validator Validating the tickets... +0ms
validator Everything passed the validation on data/1.ticket +4ms
validator:error Error, total don't match... replacing with real one +3ms
validator Everything passed the validation on data/2.ticket +0ms
validator:error Error, the number of items and the number of lines doesn't match at file data/2.ticket +1ms
validator:error Fixing it with the items length +0ms
validator:error Error, total don't match... replacing with real one +0ms
validator Everything passed the validation on data/fail.ticket +0ms
validator:error Error, total don't match... replacing with real one +0ms
main Everything worked fine... the data is storing now... +12ms
```

Los mensajes vienen en el formato:

```
modulo:detalle <mensaje> +tiempo
```

Con esto se puede apreciar, que el modulo `validator` ha encontrado errores en nuestros archivos y nos avisa que por este motivo, hara una adaptación de los datos.

Cuando algo falla, nos encontraremos con mensajes encerrados:

```
[Error: Algun error]
```

```
> ticket_parser@0.0.1 start /Users/Sier/git/ticket-homerwork
> DEBUG=* node main.js "data"

main Error happend +0ms
[Error: Item found outside the items list.]
```

Base de Datos

SQLite

Por cuestiones de simplicidad y tiempos, se incluye SQLite como el manejador de base de datos, para hacer su comunicación adecuada con el proyecto se utiliza la versión 3 y se comunica mediante el modulo de sqlite3 para Node.js.

DDL

El DDL con el cual se construyo la base de datos es muy sencillo y se incluye a continuación:

```
CREATE TABLE IF NOT EXISTS product(  
  id TEXT PRIMARY KEY NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS client(  
  id TEXT PRIMARY KEY NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS ticket(  
  id TEXT PRIMARY KEY NOT NULL,  
  client_id TEXT NOT NULL,  
  total NUMBER NOT NULL,  
  ticket_date NUMBER NOT NULL,  
  currency TEXT NOT NULL,  
  FOREIGN KEY(client_id) REFERENCES client(id)  
);  
  
CREATE TABLE IF NOT EXISTS productsInTicket(  
  ticket_id TEXT NOT NULL,  
  product_id TEXT NOT NULL,  
  age NUMBER NOT NULL,  
  quantity NOT NULL,  
  value NOT NULL,  
  FOREIGN KEY(ticket_id) REFERENCES ticket(id),  
  FOREIGN KEY(product_id) REFERENCES product(id),  
  PRIMARY KEY (ticket_id, product_id)  
);
```

Ejemplos Consultas

Las consultas se crean dentro del programa, estas son llenadas dinamicamente mediante el modulo de SQLite3, la manera es creando "statements" que se llenan con objetos.

A continuación se muestran las consultas:

```
--- Ingresar un nuevo producto
INSERT OR IGNORE INTO product VALUES(?)

--- Ingresar un nuevo cliente
INSERT OR IGNORE INTO client VALUES(?)

--- Ingresar o actualizar un ticket
INSERT OR REPLACE INTO ticket VALUES($id,$client_id,$total,$ticket_date, $currency)

--- Ingresar o actualizar los productos que tiene un ticket
INSERT OR REPLACE INTO productsInTicket VALUES($ticket_id, $product_id, $age, $quantity,
```

En los casos de las sentencias donde se incluye el "?" este caracter es reemplazado en el programa con un unico valor, en los demas casos se reemplazan los parametros con `$texto` por un valor encontrado dentro de un objeto.