**Diseño Técnico Proyecto BBVA Practitioner**

1. Introducción.

Se inicia con la arquitectura general del proyecto realizado, así como e entorno donde está desplegado y sus componentes.

También donde se especifica a detalle las estructuras internas del aplicativo back y front del proyecto realizado.

Se describe los servicios REST expuestos y la API tercera consumida.

Presenta los enlaces de ruteo de cada componente creado en el front end.

Por último, una revisión de la aplicación de sms consumida como API.

[1) Introducción. 1](#_Toc535832159)

[2) Arquitectura del proyecto 3](#_Toc535832160)

[3) Arquitectura frontend 4](#_Toc535832161)

[3.1. Estructura de carpetas. 4](#_Toc535832162)

[3.2. Estructura de navegabilidad. 4](#_Toc535832163)

[3.3. Librerías. 5](#_Toc535832164)

[3.4. Package.json 5](#_Toc535832165)

[4) Arquitectura backend 6](#_Toc535832166)

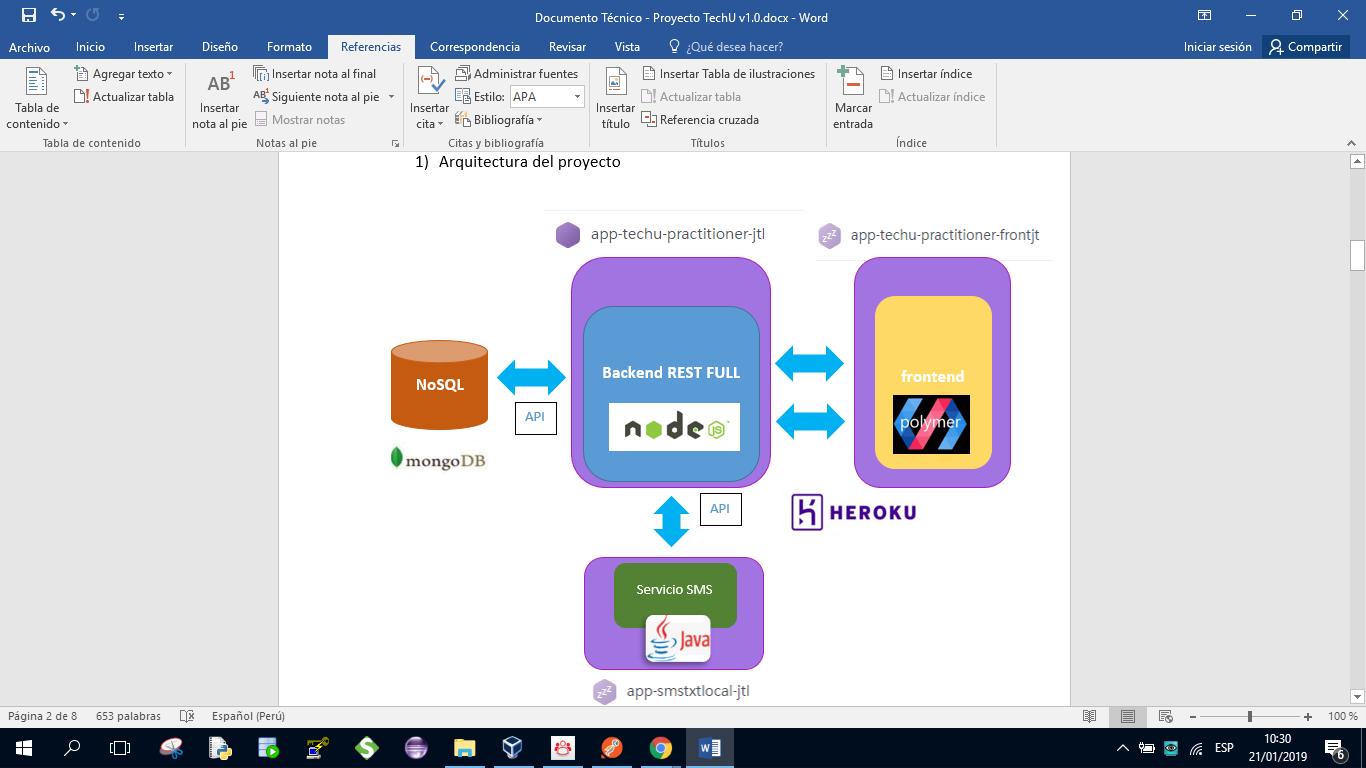
[4.1. Estructura de carpetas. 6](#_Toc535832167)

[4.2. Librerías destacadas. 6](#_Toc535832169)

[4.3. API REST FULL 7](#_Toc535832170)

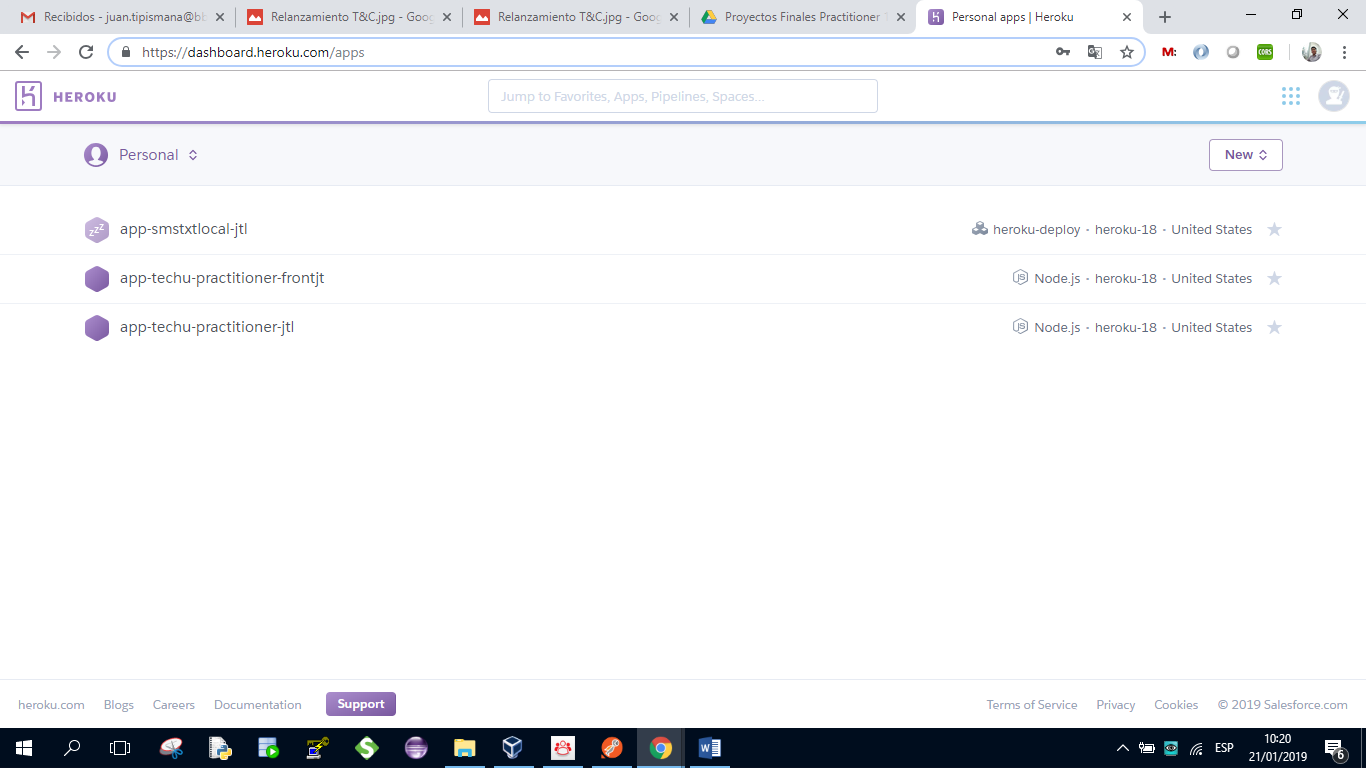
[5) API envío sms 9](#_Toc535832171)

1. Arquitectura del proyecto



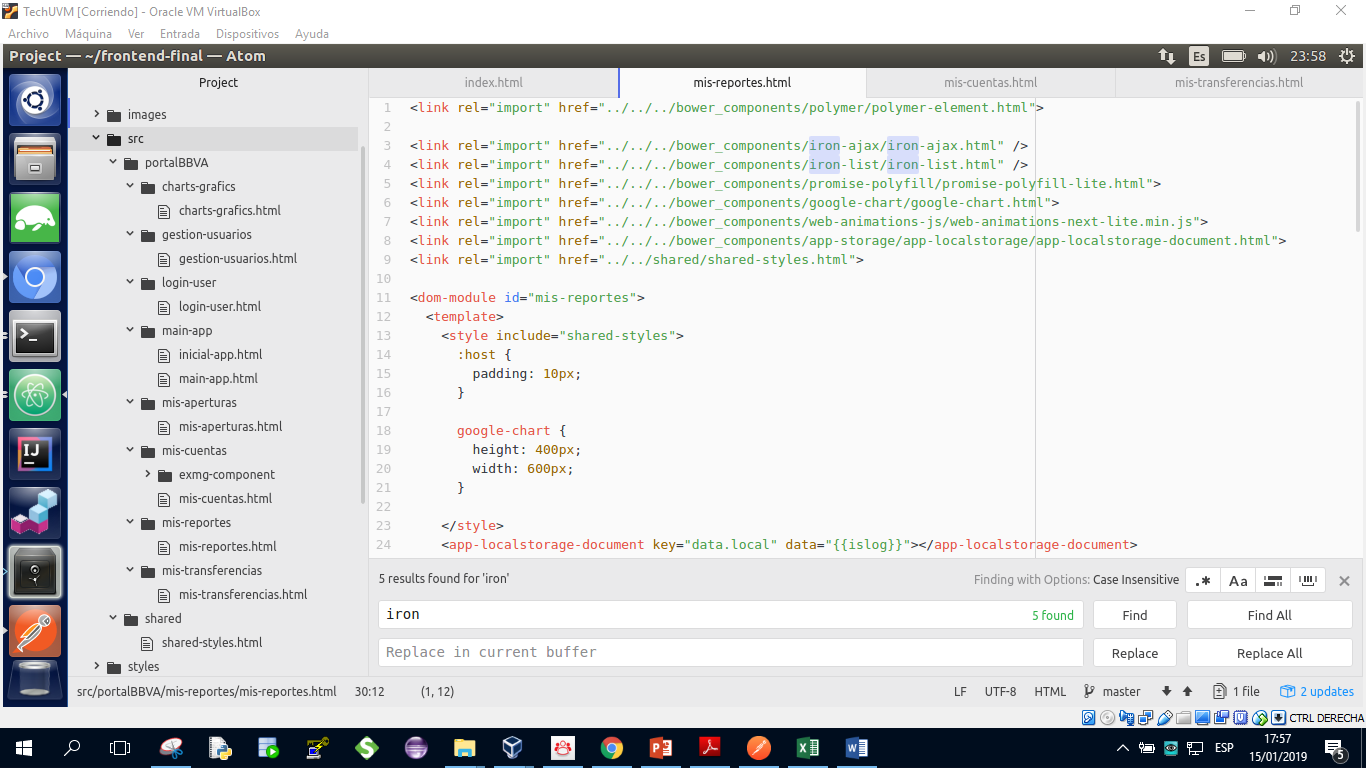
PaaS: Heroku

Se muestran las instancias de cada aplicación en la nube de Heroku.



1. Arquitectura frontend

3.1. Estructura de carpetas.

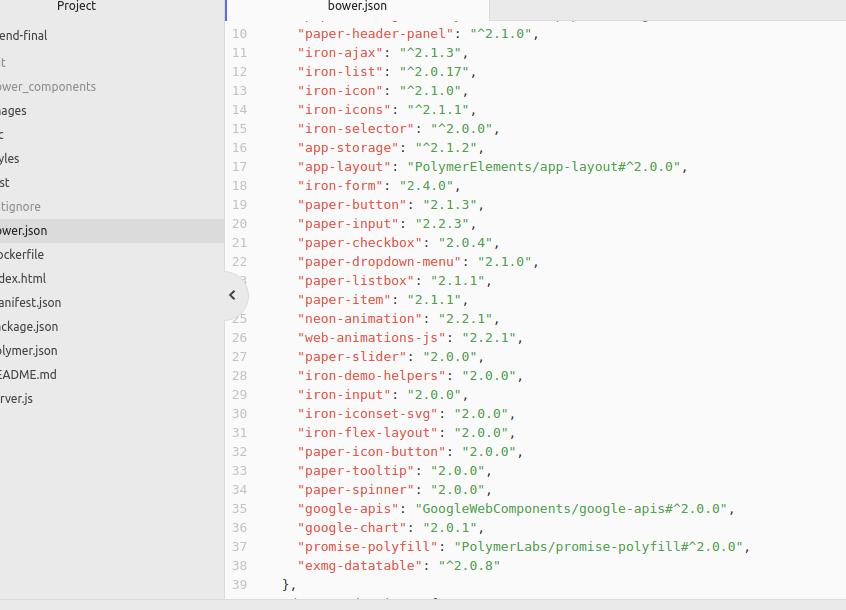


* Arquitectura Front end basada en Web Components dentro de los cuales tenemos los siguientes componentes:
* Componente Gráfico.
* Componente de Gestión de Usuarios.
* Componente login.
* Componente Principal.
* Componente de Mis Aperturas.
* Componente de Mis cuentas.
* Componente de Mis Reportes.
* Componente de Mis Transferencias.

3.2. Estructura de navegabilidad.

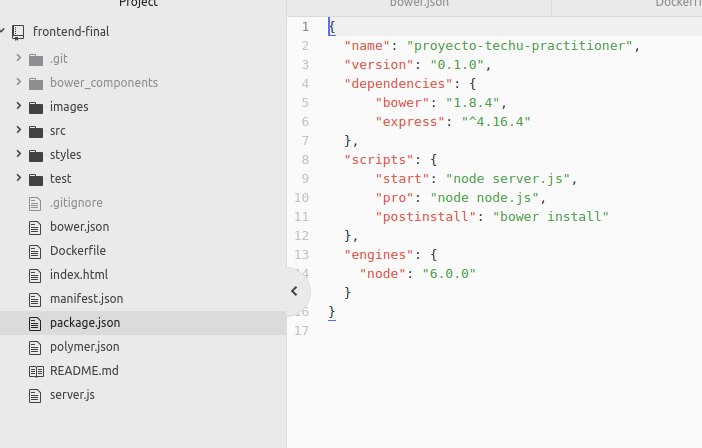
|  |  |  |  |
| --- | --- | --- | --- |
| Módulo | Method | id | url |
| login | POST | doUserPost | [[api.api\_host]][[api.api\_users]] |
| Gestión-usuarios | GET | doUserGet | [[api.api\_host]][[api.api\_users]]/[[islog.userID]] |
| PUT | doUserPut | [[api.api\_host]][[api.api\_users]]/[[islog.userID]] |
| DELETE | doUserDelete | [[api.api\_host]][[api.api\_users]]/[[islog.userID]] |
| Main | POST | doLogout | [[api.api\_host]][[api.api\_logout]] |
| POST | doLogin | [[api.api\_host]][[api.api\_log]] |
| Mis-aperturas | PUT | doAccountPut | [[api.api\_host]][[api.api\_users]]/[[islog.userID]]/[[api.api\_accounts]] |
| Mis-cuentas | GET | getAccount | [[api.api\_host]][[api.api\_users]]/[[islog.userID]] |
| Mis-reportes | GET | getReporting | [[api.api\_host]][[api.api\_users]]/[[islog.userID]]/[[api.api\_reportingManager]] |
| Mis-transferencias | PUT | other | [[api.api\_host]][[api.api\_otherTransaction]] |
| POST | postEnvioSMS | [[api.api\_host]][[api.api\_users]]/[[api.api\_enviosms]] |
| GET | getAccountDropdown | [[api.api\_host]][[api.api\_users]]/[[islog.userID]]/[[api.api\_accountDropdown]] |
| PUT | putAccountTransfer | [[api.api\_host]][[api.api\_users]]/[[islog.userID]]/[[api.api\_accountTransaction]] |

3.3. Librerías.



*Librerías destacadas:*

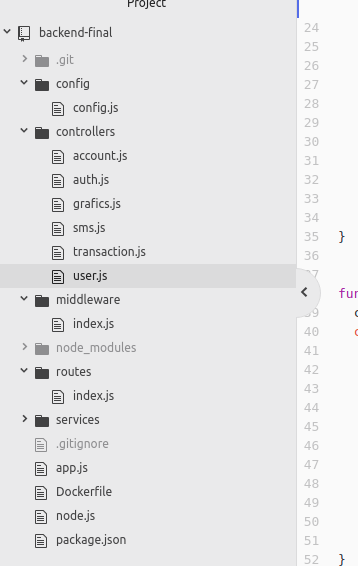
* **google-chart:** Un tipo Librería de google (de muchos) usado para el reporte gráfico de barras.
* **google-apis:** Librería complementaria que aporta en estilos, colores y tamaño a **google-chart.**
* **Exmg-datatable:** Librería para mostrar un datatable dinámico y ligero.
* **app-storage:** Librería para almacenamiento en memoria local (sesiones).
* **paper-\*** : Librería para compontes visuales.
* **iron-\*:** Librería para compontes visuales.

3.4. Package.json

* Necesario para deplegar en la nube de Heroku ya que se despliega como proyecto node js, se lee el comando “start”, y ejecuta lo que contiene (archivo server.js).
* Luego con el comando “postinstall” se instala las librerías del gestionador de librerías bower y sus dependencias.

1. Arquitectura backend

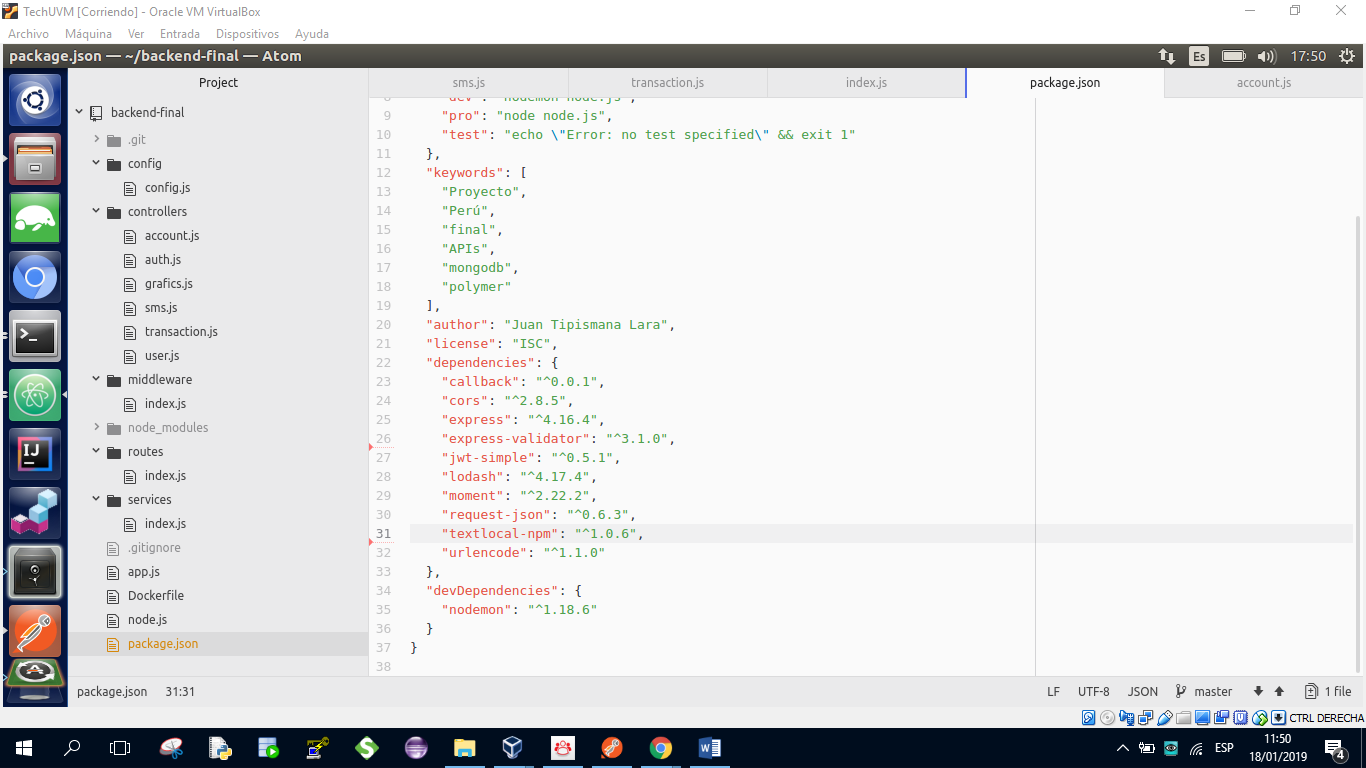
4.1. Estructura de carpetas.



*Arquitectura en capas:*

* **Config**: Contiene ApiKey de mlab, secret\_token, puerto, nombre de bd, ruta de mlab host.
* **Controllers**: Controller de cuentas (encargado de las operaciones GET, PUT, DELETE sobre las cuentas), Controller Auth (para controlar el login y logout), Controller Grafics ( encargado de mostrar el reporte gráfico de ingresos y egresos), Controller sms (encargado de conectarse a la API externa “app-smstextlocal-jtl” para el envío de mensajes de texto.), Controller transaction (encargado de las operaciones GET, PUT, DELETE sobre las tx), Controller User (encargado de las operaciones GET, PUT, POST, DELETE sobre los usuarios).
* **Routes**: Gestiona las llamadas a los controllers.
* **Services**: Controla el acceso al sistema mediante el créate token y decode token

4.2. Librerías destacadas.



**Jwt-simple**: Librería para creación de token security.

**textlocal-npm**: Librería para el envío de sms.

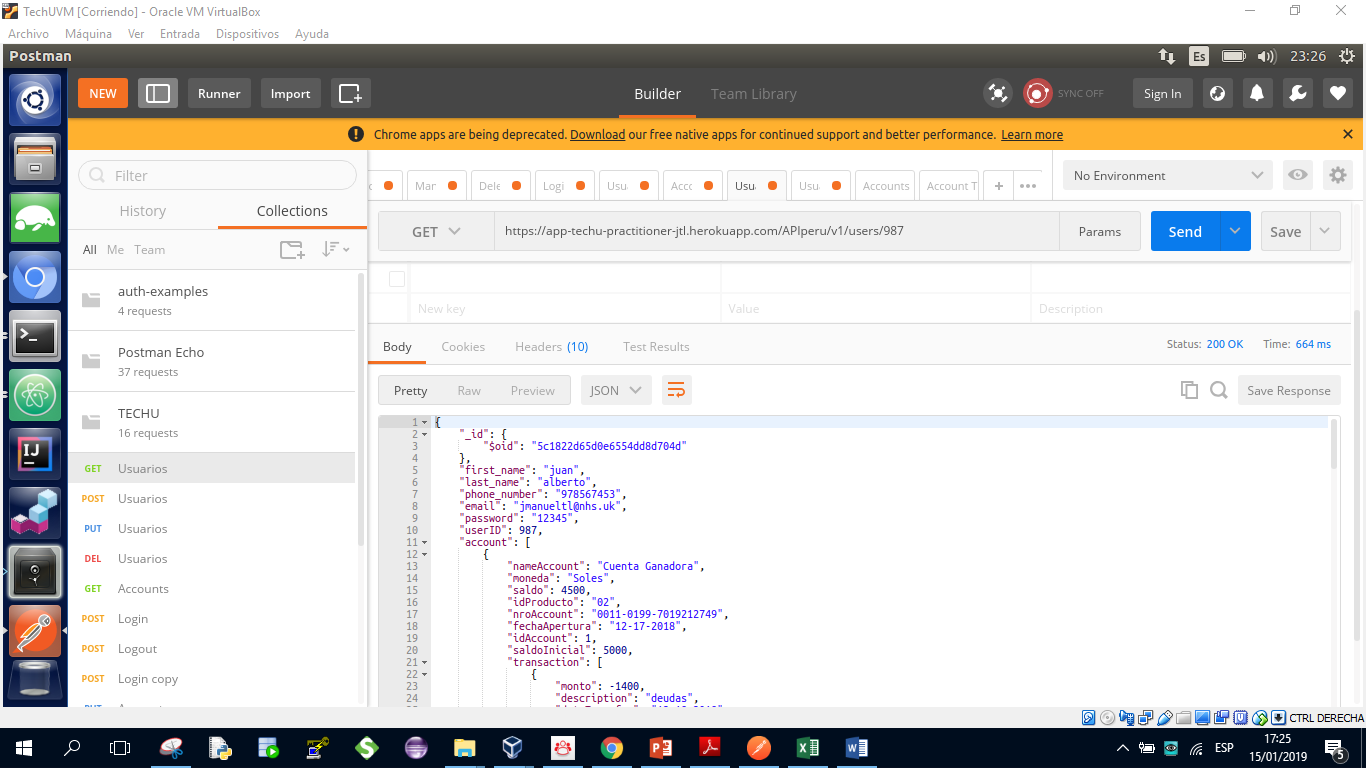
**Moment**: Librería para manejo de fechas, horas y calendarios.

**Express:** Framework para nodejs que trabaja con JavaScript

4.3. API REST FULL

|  |  |  |  |
| --- | --- | --- | --- |
| Módulo | Método | Definición Servicio REST | Ruteo |
| User | GET | Obtiene usuario determinado | /users/:id |
| GET | Obtiene todos los usuarios | /users |
| POST | Crea un nuevo usuario | /users |
| PUT | Modifica usuario existente | /users/:id |
| DELETE | Elimina usuario existente | /users/:id |
| Accounts | GET | Obtiene las cuentas del usuario | /users/:id/accounts |
| GET | Muestra las cuentas en el listado de cuentas disponibles | /users/:id/dropdownaccount |
| PUT | Modifica una cuenta determinada | /users/:id/accounts |
| DELETE | Elimina una cuenta determinada | /users/:id/accounts/:idAccount |
| Transaction | PUT | Crea una transacción entre mis propias cuentas, modificando el documento del usuario. | /users/:id/transaction |
| DELETE | Elimina una transacción determinada de un usuario. | /users/:id/accounts/:idAccount/transaction/:idTransaction |
| PUT | Crea una transacción a otra cuenta modificando el documento del usuario. | /transactionaccount |
| sms | POST | Envío de confirmación de la transacción realizada. | /users/enviosms |
| grafics | GET | Obtiene el listado de movimientos (ingresos y egresos) para mostrarlos en el gráfico de barras. | /users/:id/managereporting |
| login | POST | Crea una instancia de una sesión y un token | /login |
| logout | POST | Cierra la instancia de la sesión creada y el token. | /logout |

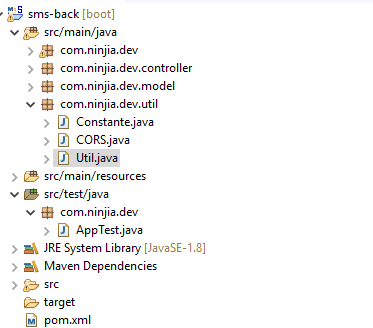
Servicios desde Postman



1. API envío sms

Se muestra la estructura de carpetas de la aplicación de envío sms

Sms-app



Se requiere las siguientes variables que se muestran definidas en la siguiente imagen para consumir el servicio cuando esté expuesto.