|  |  |  |
| --- | --- | --- |
|  |  | Informe final de entrega  SOLUCIÓN PARA QUE LOS CONTRIBUYENTES PUEDAN EMITIR UN COMPROBANTE DE PAGO Y GENERAR INFORMACIÓN DE GESTIÓN |

Solución desarrollada en la Hackathon de la SUNAT que quedo en segundo puesto.

**Orden de Servicio N° 2018S04590**

Lider del equipo:

Juan José Miranda del Solar

# contexto

La presente solución fue desarrollada en el marco de la Hackathon convocada por la SUNAT, evento en el cual se convoca a programadores para desarrollar soluciones informáticas de forma colaborativa en un plazo de 24 a 48 horas como parte de un equipo.

En nuestro caso nos presentamos como un equipo, siendo los integrantes los siguientes:

* Juan José Miranda del Solar - Lider
* Alonso Rejas
* Manuel Hohagen Serpa
* Victor David Ramos Brast
* Roberto Principe – Documentador

# detalle de la solución

La solución consta de los siguiente componentes:

## Componente 1

Aplicación móvil iOS y Android para ser usada por parte de los comerciantes como por parte de los clientes.

Aplicación desarrollada sobre tecnología Ionic framework – Angular

## Componente 2

Backend de la solución desarrollado sobre tecnología Python a la cual se conecta la app móvil y el chat bot desarrollado sobre la billetera Ethereum Toshi para la ejecucución de los pagos de los comprobantes.

## Componente 3:

Chat bot desarrollado en NodeJS bajo el estándar de bots de la billetera móvil de la red blockchain de Etherum llamada Toshi, tanto para iOS como para Android - ToshiAPP

## Componente 4:

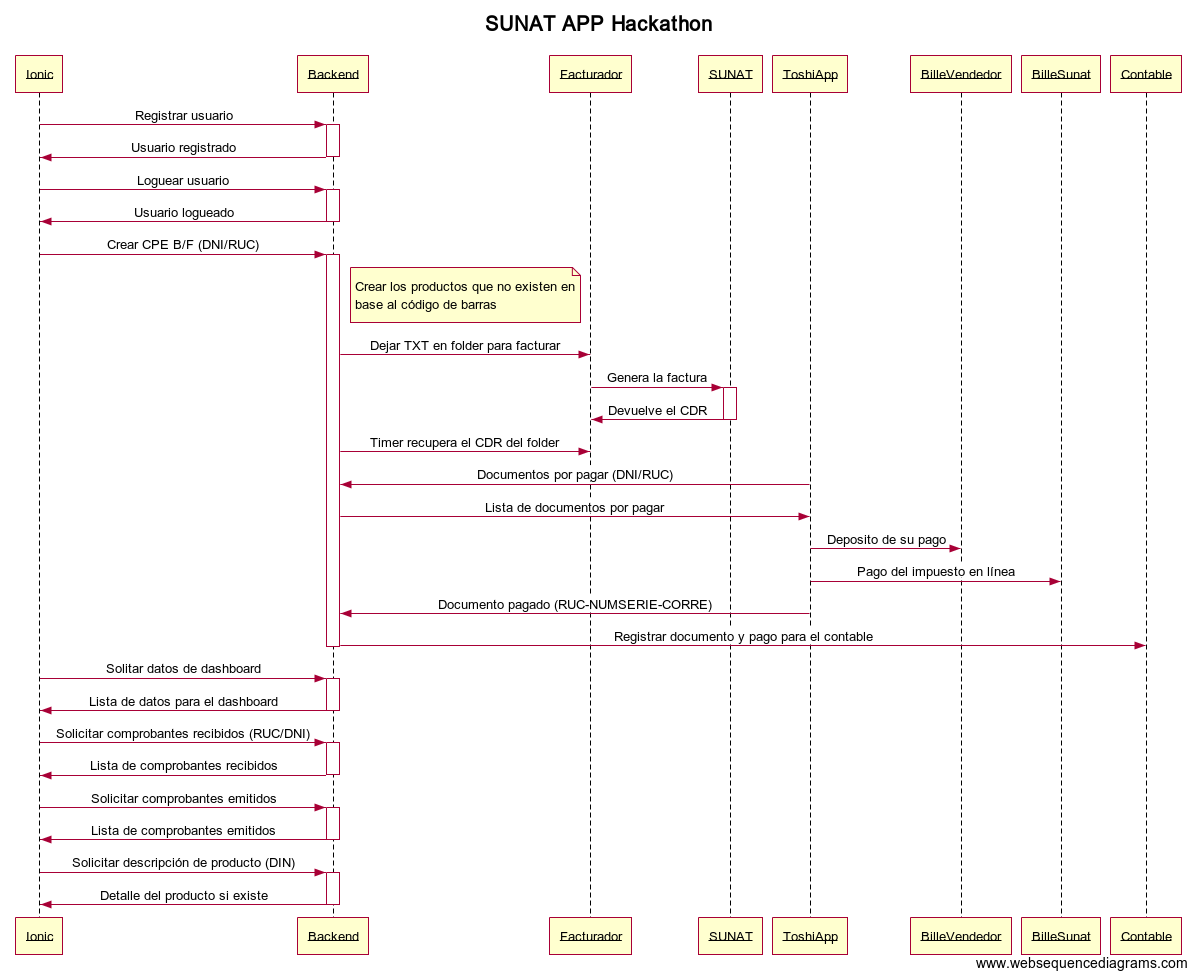
El módulo facturador que se encarga de descargar la información dejada por el backend para generar el documento electrónico contra los servicios de SUNAT. Este módulo está desarrollado en Java.

## Componente 5:

El módulo contable. Módulo que genera los libros contables electrónicos en base a la data almacenada por el backend en la base de datos de este componente.

## Diagrama de secuencia de los componentes:

Los componentes desarrollados en la hackathon actúan de la siguiente manera:



# Componente 1: App móvil

Es la aplicación móvil tanto para iOS y Android desarrollado sobre tecnología IONIC Framework & Angular.

Usa el paquete de lectura de códigos de barras de IONIC para scanear el código de los productos al generar los comprobantes de pago, el detalle y el procedimiento de instalación se encuentra documentado aquí:

<https://www.techiediaries.com/barcode-qr-code-scanner-encoder-ionic-3/>

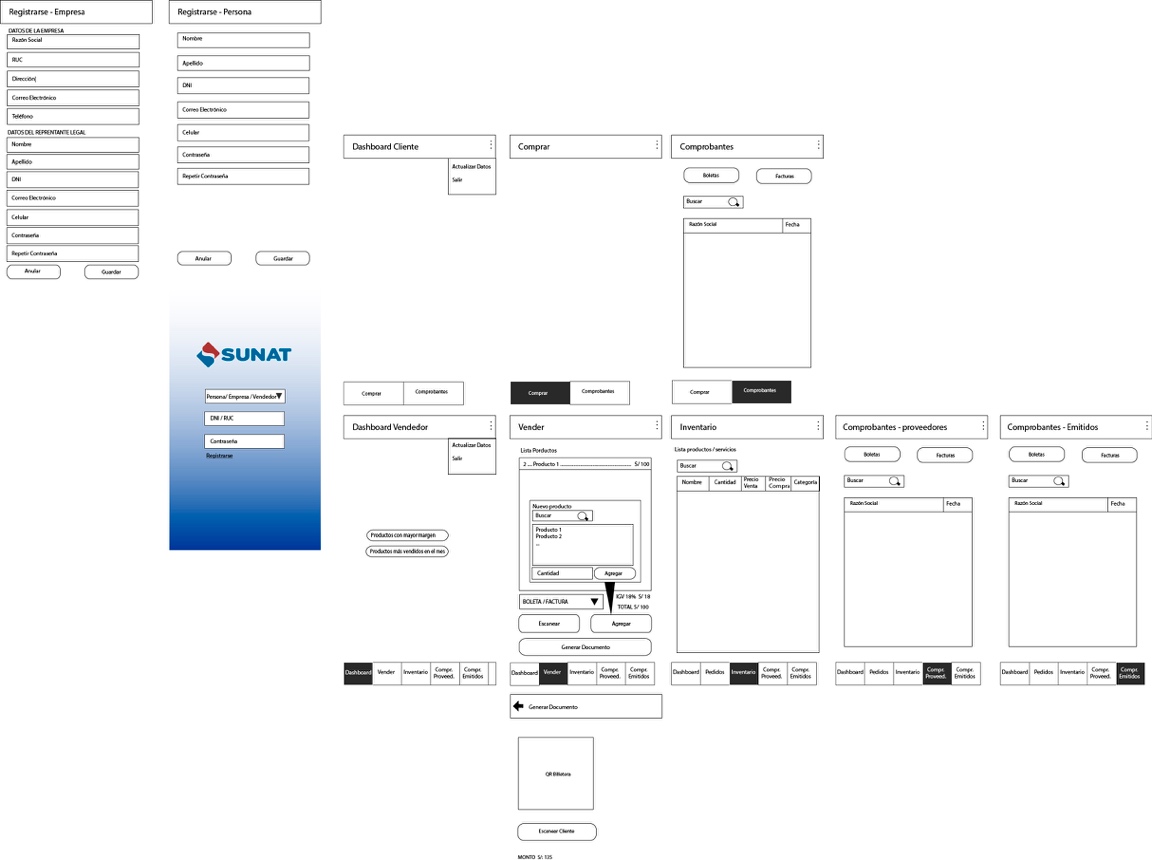
ionic cordova plugin add phonegap-plugin-barcodescanner

npm install --save @ionic-native/barcode-scanner

El código fuente del componente 1 se encuentra en este repositorio:

<https://github.com/jjmiranda/bot-hackathon-sunat/tree/master/app-frontend-ionic>

Los prototipos de las pantallas usadas son las siguientes:



El archive en formato Adobe Illustrator para cualquier modificación o mejora se encuentra disponible en la siguiente ruta:

<https://github.com/jjmiranda/bot-hackathon-sunat/blob/master/docs/Prototipos.ai>

Directorio con los códigos fuentes:

config.xml

ionic.config.json

package-lock.json

package.json

resources/

src/

tsconfig.json

tslint.json

typings/

typings.json

# Componente 2: Backend

Es el backend integrador de todos los demás componentes de la solución.

El componente esta desarrollado como una aplicación Django sobre Python.

Este backend se conecta a tres bases de datos según el archivo de definición de setting que se encuentra en:

<https://github.com/jjmiranda/bot-hackathon-sunat/blob/master/backend2/sunat/sunat/settings.py>

La base de datos principal a la cual accede el backend es la definida en “default”, por defecto debería de ser una base de datos con una carga inicial de datos del archivo dump que se encuentra en la siguiente ruta:

<https://github.com/jjmiranda/bot-hackathon-sunat/blob/master/backend2/sqlbackup.sql>

Este dump crea y puebla la base de datos con la data inicial de la aplicación.

El usuario del Django ADMIN creado es:

Usuario: admin

Clave: Magia 2015

La base de datos “facturación” y “contabilidad” son las proporcionadas por el proveedor de factura electrónica para generar el comprobante de pago electrónico y los libros de contabilidad respectivamente.

El detalle del atributo DATABASES del archivo settings.py es el siguiente:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'sunat2',

'USER': 'usr\_sunat',

'PASSWORD': 'usr\_sun4T',

'HOST': 'localhost',

'PORT': '3306',

},

'facturacion': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'sunat-email',

'USER': 'admmagia',

'PASSWORD': 'Magia2017',

'HOST': '5.189.144.206',

'PORT': '3306',

},

'contabilidad': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'magiadb',

'USER': 'admmagia',

'PASSWORD': 'Magia2017',

'HOST': '5.189.144.206',

'PORT': '3306',

}

}

El código fuente de todo este componente lo encontramos en:

<https://github.com/jjmiranda/bot-hackathon-sunat/tree/master/backend2/sunat>

El listado del directorio es el siguiente:

manage.py

sunat/

tienda/

# Componente 3: Toshi app

Este componente es un chatbot construido según las características técnicas de una aplicación para el navegador del Blockchain Ethereum TOSHI según la documentación de desarrollo que se puede encontrar en la siguiente ruta:

<https://developers.toshi.org/docs>

La aplicación TOSHI tanto para iOS como para Android se puede descargar de la siguiente ruta:

<https://www.toshi.org/>

La arquitectura del componente Toshi APP es el siguiente:

diagram

## toshi-headless-client

Este es el cliente que provee un wrapper alrededor de los servicios backend de Toshi (similiar al cliente iOS y Android). Este maneja también toda la encriptación de los mensajes extremo-a-extremo usando el protocolo SIGNAL. Está escrito en Java y corre en el background, proxeando todos los request desde el bot y hacia el bot.

## redis

Usamos redis pub/sub para proveer la conexión entre el toshi-headless-client y el bot PAGUITO.

## bot.js

Aquí es donde está toda la lógica del bot PAGUITO SUNAT.

## postgres

La base de datos Postgres es usada para almacenar toda la data de las sessiones con los usuarios, por lo cual se convierten en estados persistentes para cada usuario con quien el bot habla (similar a los cookies in un web browser), para que el bot interactue de una mejor manera con los usuarios que ya identificó.

Las documentación que se usó para crear y desplegar la aplicación TOSHI APP chatbot se encuentra disponible en el siguiente enlace:

<https://developers.toshi.org/docs/creating-a-token-app>

Crear un docker-compose.yml al momento de desplegar el chatbot según la documentación y el seed generator para proteger las 12 palabras claves a las que se asocia la billetera pública y la llave privada que manejar el chatbot en donde podrían llegar el pago de los impuestos y el pago al comerciante directamente desde la billetera Toshi del cliente.

El código fuente de todo este componente lo encontramos en:

<https://github.com/jjmiranda/bot-hackathon-sunat/tree/master/bot-sunat-toshiAPP>

El listado del directorio es el siguiente:

Dockerfile

LICENSE

Procfile

app.json

attachments/

bin/

build.gradle

config/

currency.json

dependencies/

docker-compose.yml.sample

examples/

gradle/

gradlew\*

gradlew.bat

package.json

src/

system.properties

yarn.lock

# Componente 4: Facturador

Es el componente responsable de la generación de los documentos electrónicos y quien se conecta con los servicios de la SUNAT para firmarlos digitalmente y generar el documento electrónico valido.

El componente esta construido sobre JAVA.

El código fuente de todo este componente lo encontramos en:

<https://github.com/jjmiranda/bot-hackathon-sunat/tree/master/FacturadorSunat>

El listado del directorio es el siguiente:

.classpath

.project

.settings/

WebContent/

build/

src/

# Componente 5: contable

Es el componente que en base a los documentos generados por el Backend y pagados desde el Toshi APP genera los libros contables.

El componente esta desarrollado íntegramente como PERL Modules.

El código fuente de todo este componente lo encontramos en:

<https://github.com/jjmiranda/bot-hackathon-sunat/tree/master/contable>

El listado del directorio es el siguiente:

modulos/

sql/

wpos/