PRACTICUM 1. Software developers internship en aplicación fintech

Proyecto. “Software developers internship de la aplicación Nerito”

Integrantes
Alejandro Lima Martinez (00346649)
Emma Álvarez Félix (00335568)

Asesor: Roberto Landeta Rojas

Sinodales:

Maria Del Carmen Villar Patiño
Miguel Angel Mendez Mendez
Maria Teresa Inestrillas Zarate

Glosario

- Figma: Es un programa que ofrece todas las herramientas necesarias para diseñar un proyecto. Sobre todo es ideal para crear interfaces de usuario tanto para web como para móvil.
- GitLab: Es una aplicación basada en la web con una interfaz gráfica de usuario que también puede instalarse en un servidor propio. El núcleo de GitLab lo forman los proyectos en los que se guarda el código que va a editarse en archivos digitales, los denominados repositorios.
- React js: Es una biblioteca escrita en JavaScript, desarrollada en Facebook para facilitar la creación de componentes interactivos, reutilizables, para interfaces de usuario.
- Request: Permite el acceso a toda la información que pasa desde el navegador del cliente al servidor.
- API: Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones.
- JSON: Cuyo nombre corresponde a las siglas JavaScript Object Notation o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

Introducción

Las fintech son una industria naciente en la que las empresas usan la tecnología para brindar servicios financieros de manera eficiente, ágil, cómoda y confiable. La palabra se forma a partir de la contracción de los términos finance y technology en inglés.

Nerito es el nombre de la fintech en donde actualmente trabajamos como becarios bajo el cargo de software developers.

En el siguiente documento, presentaremos la calendarización y resultados de las tareas correspondientes a las etapas presentadas en el protocolo de la materia **Practicum I de Ingeniería en sistemas** el cual se encuentra expresado gráficamente en el apartado de anexos.

Marco Teórico

Las finanzas electrónicas han avanzado en los últimos años, con la aparición de las criptomonedas, como Bitcoin, el mercado financiero ha empezado a hacer transacciones electrónicas de todo tipo por medio de dispositivos móviles. Hay fintechs especializadas en la compra-venta de criptomonedas y otras como Nerito que permiten realizar pagos, préstamos, coperachas de manera electrónica.

Con la pandemia de COVID-19 muchos de los procesos digitales se han acelerado, especialmente la interconexión de las compras en línea así como el uso de wallets virtuales para tener dinero digital seguro y realizar compras sin tener un riesgo físico.

Según el reporte “FINTECH, REGTECH AND THE ROLE OF COMPLIANCE 2021” se ha incrementado el uso de tecnologías financieras más de un 50% en el 2020 con respecto a los años pasados, esto también provocó el surgimiento de nuevas plataformas y aplicaciones en el mercado; Actualmente en México existen aproximadamente unas 330 empresas fintech.

El lanzamiento de una fintech no depende únicamente del desarrollo y la infraestructura tecnológica, para que una aplicación fintech pueda operar legalmente en México debe de cumplir con los lineamientos establecidos por la Comisión Nacional Bancaria de Valores (CNBV), BANXICO y CONDUSEF.

Cumplir las regulaciones es necesario ya que las aplicaciones de esta índole utilizan el dinero de los usuarios, por lo que se debe garantizar la seguridad dentro de todos los movimientos. Las normas impuestas por los organismos bancarios verifican la seguridad de los datos sensibles (financieros y personales) ante ataques cibernéticos (hackeos), físicos (robos) y desastres naturales que pudieran afectar al cliente y su dinero.

Las transacciones monetarias de una fintech en el país se monitorean por el Sistema de Pagos Electrónicos Interbancarios (SPEI). El sistema es la infraestructura de pagos del Banco de México que permite a sus participantes, (bancos, casas de bolsa, sofipos y otras entidades financieras reguladas) enviar y recibir pagos entre sí para poder brindar a sus clientes finales el servicio de transferencias.

Las transacciones de SPEI generan un archivo .txt que contiene datos confidenciales del usuario que ha realizado una transacción, estos datos pasan por un proceso de encriptación para evitar el robo de información y garantizar su seguridad, una vez que los datos se guardan en la base datos dentro del archivo .txt pueden ser utilizados para reflejar al usuario su estado financiero y generar analíticos a la fintech.

Objetivos específicos

- Mostrar los conocimientos obtenidos en la universidad con la participación en la empresa Nerito tales como: **Desarrollo de aplicaciones móviles, Calidad y pruebas de software, Desarrollo de aplicaciones web, Bases de datos 1, Bases de datos 2, Sistemas Operativos**.
- Enseñar fragmentos de código, explicar la lógica y funcionalidad, mostrar visualmente los resultados del código (front-end[1]).

Métododologia[2]

La metodología a seguir para la realización de nuestro respectivo trabajo se divide en etapas, adicionalmente se tienen juntas diarias llamadas dailies con una duración de 15 minutos, en las cuales se establecen futuras tareas, se comentan los avances y posibles obstáculos de las tareas en desarrollo.
Las etapas de la metodología son:

1. Entendimiento del requerimiento:

   En esta etapa nos dedicamos a entender la problemática expuesta por el project manager, en caso de que la situación sea de un entendimiento abstracto, solicitaremos apoyo de los ingenieros seniors para establecer los parámetros para solucionar la problemática en su totalidad.

2. Preparación de los materiales de trabajo:

   Se clonan y actualizan los repositorios, documentos o herramientas necesarias para llevar a cabo la solución del problema.

3. Planteamiento de la solución:

   Analizamos la problemática con mayor profundidad y con base en las herramientas existentes, desarrollamos el planteamiento de una solución que cumpla con los requerimientos en su totalidad.

4. Desarrollo de la solución:

   En esta etapa se lleva a cabo el desarrollo y diseño (en caso de ser necesario) de la solución, esta etapa puede incluir las actividades de codificación, testeo, diseño y redacción, según sea el planteamiento establecido en la etapa anterior.

---

[1] diseño de un sistema, desde la estructura de este hasta los estilos como colores, fondos, tamaños, animaciones y efectos.
[2] Al ser becarios y tener tareas cambiantes, cada quién mostrará en lo que trabajó durante el periodo de Septiembre-Octubre.

5. Presentación de la solución de los cambios realizados:

   La solución que se ha llevado a cabo, se presenta ante los desarrolladores seniors o el project manager dependiendo el tipo de trabajo, quienes validan que el contenido cumpla con los requerimientos especificados.

6. Publicación de la solución:

   Una vez que el nuevo código/datos han sido revisados y aceptados por el grupo senior, podemos realizar una fusión comúnmente llamada “merge” con los cambios. En caso de la redacción guardar en el documento oficial los cambios.

Resultados y discusión

Alejandro:

- **Desarrollo de rediseño de consola de transacciones SPEI (20-09-21 - 21-09-21)**

1. El requerimiento solicitado es: Desarrollo del rediseño en figma de la consola de transacciones de SPEI donde se integre un apartado que muestre el número de transacciones, número de usuarios que realizaron una transacción, cantidad total de transacciones y fecha del registro del documento de transacciones SPEI.
2. Se clonó el repositorio correspondiente a la aplicación web que contiene la consola SPEI antigua y se creó una sesión en figma para su uso.
3. Se establecieron los parámetros que debe de contener la nueva consola SPEI así como los colores y las formas de los renglones y columnas.
4. Se diseñó la consola SPEI en figma basados en la etapa anterior.
5. Se establece una comunicación con el project manager quien aceptó los cambios
6. Se establecieron los cambios en el documento de figma.

Codificación de la consola SPEI actualizada (22-09-21 - 24-09-21)

1. Se requiere codificar la consola de SPEI diseñada en figma en el lenguaje de programación Java Script con el uso del framework React js.
2. Se actualizó el repositorio para tener los cambios recientes correspondientes a la aplicación web con el comando **git pull origin develop.**
3. Se revisó el diseño de figma para identificar los elementos a renderizar en la pantalla a través de código.
4. Se realizó el código correspondiente para la realización de la consola SPEI. En este punto se trató el desarrollo sin conexión a la base de datos, solo se renderiza con datos falsos para mostrar que es lo esperado según el diseño de figma, se presentaron algunos cambios sobre el diseño, donde se consideraban mejores estilos y se aprovechó para realizarlo, al ser presentado en un daily el project

manager actualizo los cambios y pidió que se guardará la imagen de figma anterior, para el control de versiones.
5. El desarrollador senior Ivan Sodari y el project manager Marco Perez quienes lo aprobaron.
6. Se hizo un merge con los cambios en gitlab para guardarlos en la versión más actualizada del repositorio.

- **Programar un crontab que genere los archivos transacciones SPEI cada 30 minutos (27-09-21 - 27-09-21)**

1. Se requiere realizar un crontab en el archivo del backend, especialmente en el archivo serverless de speis.
2. Se llevó a cabo el proceso de clonado del repositorio de gitlab para poder modificarlo así como la verificación de las credenciales de acceso a la consola de aws. Adicionalmente, se realizó la descarga de la aplicación POSTMAN para realizar los request locales.
3. Se revisó la documentación para comparar la generación de un crontab en linux y un crontab para servicios de aws.
4. Se agregó un evento a la función serverless llamada “buildSPEIfiles” existente en el archivo serverless.yaml, el nombre del evento es “schedule” y dentro de con la programación del crontab “cron(0/30 * ? * MON-FRI *)”. Se realizaron pruebas durante el día, para ver que se generen cada 30 min efectivamente siempre y cuando se presenten transacciones diferentes en el lapso de tiempo.
5. El desarrollador senior Ivan Sodari y el project manager Marco Perez quienes lo aprobaron.
6. Se subieron los cambios correspondientes al repositorio.

- **Generar una tabla con que contenga la información de cantidad total de transacciones, usuarios que realizaron transacciones y la fecha y hora en la que se generó el registro según el archivo SPEI.**

1. Se analizó la composición de las tablas almacenadas en dynamodb de aws, para identificar el nombre de la nueva, así como las columnas que llevaría.
2. Se realizó la actualización del repositorio en cuestión, para mantener todos los cambios del backend acordes a los posibles cambios hechos por el equipo de Nerito.
3. Se revisó el archivo serverless.yaml dentro del folder de speis, el cual contiene la creación de las tablas involucradas en estos procesos transaccionales, la intención de hacer es para identificar dónde hacer la tabla y basado en las tablas anteriores, generar la creación de la misma, así como el enviroment, el cual nos ayuda a hacer el llamado de la misma en otros documentos, referenciados con los datos necesarios. Adicional, también se agregó su respectivo resource, quien hace el llamado a la tabla y base de datos en dynamodb.
4. Se codificó la creación de la tabla, así como los campos que contendrá, el nombre y las propiedades de la misma. Después de ello se generó en enviroment para hacer el llamado de la misma a otros archivos para almacenar los datos una vez que se generó la tabla. Por último se agregó el resource a los statements para hacer la conexión entre dynamodb y mis archivos serverless o js para poder generar los request a la tabla.

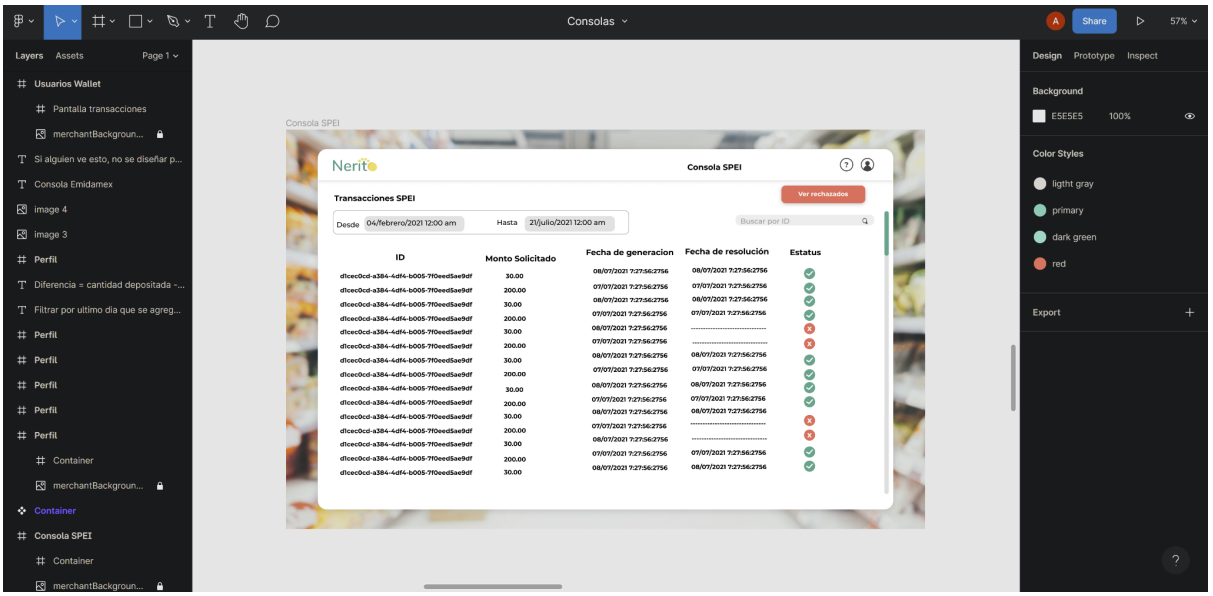
5. Dentro de este proceso el ingeniero Ivan Sodari apoyó y supervisó las actividades correspondientes a la creación de la tabla y fue aprobado por el mismo.
6. Se subieron los cambios correspondientes al repositorio.

- **Crear endpoint para reproceso de transacciones y reflejar los analiticos en la tabla creada (04-10-21 - 07-10-21)**

1. Se identificó y revisó el archivo buildSpeiFiles.js dentro de la carpeta de speis, el cual genera los archivos SPEI, los cuales contienen los datos de las transacciones de los clientes. Dichos archivos se obtienen por medio de la aplicación NERITO, por lo que corresponde a un archivo del backend. También se identificó que el query que será enviado a dynamodb se realizará en app.js.
2. Se realizó la actualización del repositorio en cuestión, para mantener todos los cambios del backend acordes a los posibles cambios hechos por el equipo de Nerito.
3. Primero se identificó la diferencia entre la generación de una transacción (archivo 04) y una alta de cuenta (archivo AC) ya que en build Speed Files.js llegan ambos y se genera el documento correspondiente.
4. Se hizo la creación de variables globales, las cuales actúan como contadores para determinar el número de usuarios y la cantidad total de transacciones, dentro de este archivo se realiza el put, el cual es el método que registra la tupla que se generó con las variables para que se establezca en la base de datos. Dentro del archivo app.js se hizo el query de get, debido a que este archivo es el que se conecta con el front-end para reflejar los registros en pantalla. Por ello después de realizar el get, se modificó el front-end, dentro del front-end realice unos cuantos cambios, primero con la API me conecto al archivo de app.js del backend, donde hago el llamado al enviroment que se creó en las tareas anteriores, para extraer los datos con el get. El objeto que devuelve, se renderiza dependiendo el estado de hook que muestra, las transacciones o los analiticos, por medio de métodos como filter, slice, stringfy, etc. transformó el JSON a un objeto iterable en js y se puede desplegar los datos.
5. Dentro de este proceso el ingeniero Ivan Sodari supervisó la realización de los procesos para el funcionamiento de los queries así como de la codificación en front-end, la cual se presentó a Marco Perez, quien la aceptó, mencionando que se le dará seguimiento en cuanto exista una conexión con Banorte para validar los SPEIS.
6. Se subieron los cambios a gitlab.

EVIDENCIA:

Diseño en figma de consola SPEI



Codificación y modificación de la consola de SPEI con extracción de datos en la tabla creada en la tarea mencionada.

```
import React from 'react';
import { useEffect, useRef } from 'react';
import { makeStyles } from
'@material-ui/core/styles';
import { withSnackbar } from 'notistack';
import CircularProgress from
'@material-ui/core/CircularProgress';
import { Typography } from
'@material-ui/core';
import { API } from 'aws-amplify';
import Pagination from
'@material-ui/lab/Pagination';
import TextField from
'@material-ui/core/TextField';
import moment from 'moment-timezone';
import Button from
'@material-ui/core/Button';
import csvtojson from 'csvtojson';
import Input from
'@material-ui/core/Input';
import InputLabel from
'@material-ui/core/InputLabel';
import FormControl from
'@material-ui/core/FormControl';

import '../../../assets/css/main.css';
import iconUpload from
'../../../assets/images/upload.svg'
import iconDownload from
'../../../assets/images/download.svg'
import noMovements from
'../../../assets/images/no_movements.svg'
import engine from
'../../../assets/images/engine.svg'
import DeafultSearch from
'../../common/deafultSearch'
import CheckValue from
"../../../lib/formatValidations";
import Modal from
'@material-ui/core/Modal';
import AyudaModal from './helpUpload';
import {FONT_DEFAULT} from
'../../../lib/constants.js'
const axios = require('axios');

function getModalStyle() {
  return {
    top: `50%`,
    left: `50%`,
    transform: `translate(-50%, -50%)`,
  };
}

const useStyles = makeStyles({
  root: {
    display: 'flex',
    flex: 1,
    backgroundColor: '#F6F6F6',
    flexDirection: 'column',
    width: '100%',
    height: '100%'
  },
  container: {
    flex: 8,
    marginLeft: 30,
    marginRight: 40,
    marginBottom: 40,
  },
  titleConteiner: {
    padding: 10,
    marginTop: 20,
    marginBottom: 30,
    flexDirection: 'row'
  },
  transactionsConteiner: {
    backgroundColor: '#FFFFFF',
    padding: '40px 50px 50px 40px',
    borderRadius: 5,
    border: '1px solid #E0E0E0'
  },
  tableContainer: {
    display: 'flex',
    marginLeft: 0,
    marginRight: 0,
    marginTop: 20,
    marginBottom: 20,
    backgroundColor: '#F9F9F9',
    borderRadius: 5,
    flexDirection: "row"
  },
  buttonContainer: {
    display: 'flex',
    marginBottom: 15,
  },
  tableGridContainer: {
    width: 'calc(100vw - 87px)',
    width: '100%',
    marginLeft: 0,
    marginRight: 0,
    marginTop: 30,
    marginBottom: 20,
  },
  actionButonsContainer: {
    display: 'flex',
    flexDirection: 'row',
    justifyContent: 'center',
    height: '100%',
    alignItems: 'center',

  },
  actionButon1: {
    cursor: 'pointer',
    marginRight: 10
  },
  actionButon2: {
    cursor: 'pointer',
    marginLeft: 10
  },
  tableHeaderCenter: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 700,
    fontSize: 14,
    display: 'flex',
    alignItems: 'center',
    color: '#434343',
    borderTop: '1px solid #BDBDBD',
    borderBottom: '1px solid #BDBDBD',
    height: 60,
    justifyContent: 'center',
    paddingLeft: 3,
    paddingRight: 3
  },
  tableHeaderText: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 700,
    fontSize: 14,
    display: 'flex',
    alignItems: 'center',
    color: '#434343',
    borderTop: '1px solid #BDBDBD',
    borderBottom: '1px solid #BDBDBD',
    height: 60,
    justifyContent: 'flex-start',
    paddingLeft: 3,
    paddingRight: 3
  },
  tableHeaderNumber: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 700,
```

```
    fontSize: 14,
    display: 'flex',
    alignItems: 'center',
    borderTop: '1px solid #BDBDBD',
    borderBottom: '1px solid #BDBDBD',
    height: 60,
    justifyContent: 'flex-end',
    paddingLeft: 3,
    paddingRight: 3
  },
  tableRowCenter: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: '600',
    fontSize: 12,
    paddingTop: 15,
    paddingBottom: 15,
    display: 'flex',
    alignItems: 'center',
    height: 10,
    textAlign: 'center',
    justifyContent: 'center',
    paddingLeft: 3,
    paddingRight: 3
  },
  tableRowText: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: '600',
    fontSize: 12,
    paddingTop: 15,
    paddingBottom: 15,
    display: 'flex',
    alignItems: 'center',
    height: 10,
    textAlign: 'center',
    justifyContent: 'flex-start',
    paddingLeft: 3,
    paddingRight: 3
  },
  tableRowNumber: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: '600l',
    fontSize: 12,
    paddingTop: 15,
    paddingBottom: 15,
    display: 'flex',
    alignItems: 'center',
    height: 10,
    textAlign: 'center',
    justifyContent: 'flex-end',
    paddingLeft: 3,
    paddingRight: 3
  },
  buttonStyle: {
    border: '1px solid #3B9A83',
    boxSizing: 'border-box',
    borderRadius: 5,
    width: 146,
    height: 38,
    margin: 2
  },
  buttonText: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 10,
    color: '#3B9A83'
  },
  plusIcon: {
    marginRight: 5
  },
  paper: {
    position: 'absolute',
    width: 497,
    height: 'auto',
    backgroundColor: '#F9F9F9',
    boxShadow: '0 4 14 rgba(0, 0, 0,
0.15)',
    borderRadius: 5,
    textAlign: 'center'
  },
  mainContainerRegister: {
    display: 'flex',
    alignItems: 'center',
    flexDirection: 'column',
    justifyContent: 'center',
    backgroundColor: 'white',
    borderRadius: 5
  },
```

```
  modalButtons: {
    marginBottom: 20,
    marginTop: 20
  },
  backgroundOddColumn: {
    backgroundColor: '#F9F9F9'
  },
  totalsLabel: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'normal',
    fontSize: 12,
    textAlign: 'center',
    color: 'rgba(67, 67, 67, 0.6)',
    textAlign: 'left'
  },
  totalsValue: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 18,
    textAlign: 'center',
    textAlign: 'left'
  },
  totalsValueAlt: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    textAlign: 'center',
    marginLeft: 5
  },
  defaultButton: {
    border: '1px solid #3B9A83',
    boxSizing: 'border-box',
    borderRadius: 5,
    width: 204,
    height: 46,
    marginLeft: 6
  },
  settingsButtonStyle: {
    border: '1px solid #3B9A83',
    boxSizing: 'border-box',
    borderRadius: 5,
    width: 161,
    height: 46,
    marginLeft: 6
  },
  successButtonText: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 10,
    color: '#3B9A83'
  },
  backgrounOddColumn: {
    backgroundColor: '#F6F6F6',
    padding: 5
  },
  backgrounColumn: {
    backgroundColor: '#FFFF',
    padding: 5
  },
  formDate: {
    marginLeft: 5,
    marginRight: 5
  },
  depositButtonText: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 600,
    fontSize: 10,
    color: '#3B9A83',
    marginLeft: 20
  },
  margin20: {
    marginBottom: 20,
    border: '1px solid #BDBDBD',
    borderRadius: 5,
    padding: 20
  },
  title: {
    ...FONT_DEFAULT,
    marginTop: 10,
    marginBottom: 0,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 24,
    color: '#434343'
  },
  statusSending: {
```

```
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    display: 'flex',
    alignItems: 'center',
    color: '#6FCF97'
  },
  statusPending: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    color: '#EFAB23'
  },
  statusSending: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    color: '#77BEE0'
  },
  statusRefund: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    color: '#DA7B7B'
  },
  statusCompleted: {
    ...FONT_DEFAULT,
    fontStyle: 'normal',
    fontWeight: 'bold',
    fontSize: 12,
    color: '#6FCF97'
  },
  errorContainer: {
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
    justifyContent: 'center',
    marginTop: 50,
    marginbottom: 80
  },
  errorText: {
    ...FONT_DEFAULT,
    fontSize: 16,
    fontWeight: 700,
    marginTop: 20
  },
  errorText2: {
    ...FONT_DEFAULT,
    fontSize: 14,
    fontWeight: 600,
    opacity: '0.6'
  },
  paginationTAB: {
    widht: '100%',
    display: 'flex',
    justifyContent: 'center'
  },
  titleFlex: {
    display: 'flex'
  },
  paper: {
    position: 'absolute',
    width: 440,
    height: 'auto',
    backgroundColor: '#FAFAFA',
    boxShadow: '0 4 14 rgba(0, 0, 0,
0.15)',
    borderRadius: 5,
    textAlign: 'center',
    paddingTop: 50
  },
  buttonWithCaption: {
    display: 'flex',
    flexFlow: 'column',
    justifyContent: 'center'
  },
  captionForButton: {
    marginTop: '5px',
    paddingLeft: '10px',
    fontSize: 'xx-small',
    color: 'gray',
  }
});

const initialFromDate =
`${moment().format('YYYY-MM-DD')}T00:00`
```

```
const initialToDate =
`${moment().format('YYYY-MM-DD')}T23:59`
const perPage = 10

function Spei(props) {
  useEffect(() => {
    callApi(initialFromDate,
initialToDate)
  }, [])

  const classes = useStyles();

  const [modalStyle] =
React.useState(getModalStyle);
  const [isLoading, setLoading] =
React.useState(true);
  const [open, setOpen] =
React.useState(false);
  const [transactions, setTransactions] =
React.useState([]);
  const [searchedWord, setSearchedWord] =
React.useState('');
  const [fromDate, setFromDate] =
React.useState(initialFromDate);
  const [toDate, setToDate] =
React.useState(initialToDate);
  const [selectedPage, setSelectedPage] =
React.useState(1);
  const [pages, setPages] =
React.useState(0);
  const [RFC, setRFC] =
React.useState('');
  const [sourceAccount, setSourceAccount]
= React.useState('');
  const [commission, setCommission] =
React.useState(0);
  const [description, setDescription] =
React.useState('');
  const [settings, setSettings] =
React.useState(null);
  const [lastACProcessed,
setLastACProcessed] = React.useState(0);
  const [lastSpeiProcessed,
setLastSpeiProcessed] = React.useState(0);
  const [ayudaUpload, setAyudaUpload] =
React.useState(false);
  const handleAlert = (variant, message)
=> {
    props.enqueueSnackbar(message, {
variant });
  };

  const getTransactions = async (from, to)
=> {
    try {
      const getTransactions = await
API.get('speis-api', '/speis', {
        queryStringParameters: {
          since: from,
          to: to,
        },
      })
      return getTransactions
    } catch (err) {
      handleAlert('error', err.message)
      return null
    }
  }

  const getSettings = async () => {
    try {
      const settingsData = await
API.get('setting-api',
'/settings/default_spei_config')
      console.info(settingsData);
      if (settingsData.data &&
settingsData.data.length > 0 &&
settingsData.data[0].setup) {

setRFC(settingsData.data[0].setup.RFC)

setSourceAccount(settingsData.data[0].setu
p.sourceAccount)

setCommission(settingsData.data[0].setup.c
ommission)

setDescription(settingsData.data[0].setup.
description)

setLastACProcessed(settingsData.data[0].se
tup.lastACProcessed || 0)

setLastSpeiProcessed(settingsData.data[0].
setup.lastSpeiProcessed || 0)
        setSettings(settingsData)
      }
    } catch (err) {
      handleAlert('error', err.message)
    }
  }
  const getSpeiFiles = () => {
    API.get('speis-api', '/speis/batch', {
      queryStringParameters: {
        AC: lastACProcessed,
        SPEI: lastSpeiProcessed,
      },
    }).then(result => {
      if (result) {
        console.info(result);
        const i =
setInterval(downloadFiles, 1000,
[result.ACAR, result.SPEI]);
        setLocalInterval(i);
        getSettings();
      }
    }).catch(error => {
      handleAlert('error', 'Error
obteniendo archivos de transacciones
SPEI');
      console.error(error);
    })

  }
  const CargarErrores = async event => {
    const [file] = event.target.files;
    const fileNameRX =
/(ERRORES|EXITO)-(AC|SPEI)-\d+\.txt/ig;

    if (fileNameRX.test(file.name)) {
      const lote =
file.name.replace(/\w+-(\w+-\d+\.txt)/ig,
'$1');

      if (file.size === 0 &&
/EXITO/ig.test(file.name)) {
        handleAlert('info', `El 100% de
las transacciones del ${lote} serán
procesadas.`);
      } else {
        handleAlert('info', `Se procesarán
las transacciones del ${lote} como errores
y el resto como ENVIADAS.`);
      }
      const uploadUrl = await
API.get('speis-api', '/speis/results', {
        queryStringParameters: {
          file: file.name,
        },
      });
      console.info(uploadUrl);
      if (uploadUrl.data.url) {
        await
axios.put(uploadUrl.data.url, file, {
headers: { 'Content-Type': 'text/plain' }
});
      } else {
        handleAlert("error", "No se logró
subir el documento");
      }

    } else {
      handleAlert('warning', 'El nombre
del archivo de errores debe ser
Errores-AC-#.txt o Errores-SPEI-#.txt')
    }
  }
  const [localInterval, setLocalInterval]
= React.useState(null);
  const downloadFiles = (urls) => {
    try {
      if (urls.length === 0) {
        clearInterval(localInterval);
        setLocalInterval(null);
        return;
      }

      const url = urls.pop();
      console.log('downloading', url);
      let _a =
document.createElement('a');
      document.body.appendChild(_a);
      _a.setAttribute('href', url);
      _a.setAttribute('download',
url.split('?')[0].split('/').pop());
      _a.click();
      document.body.removeChild(_a);
    }
    catch (error) {
      clearInterval(localInterval);
      console.error(error);
    }
  }
  const getUsers = (userIds) => {
    return userIds.map(async userId => {
      const getUser = await
API.get('profile-api',
`/profile/${userId}`);
      return getUser.data[0]
    })
  }

  const callApi = async (from, to) => {
    try {
      const timeStampFrom =
moment.utc(from).valueOf()
      const timeStampTo =
moment.utc(to).valueOf()
      setLoading(true)
      if (settings === null) {
        getSettings()
      }
      if (timeStampFrom < timeStampTo) {
        const getTrans =
getTransactions(timeStampFrom,
timeStampTo)
        const trans = await
Promise.resolve(getTrans)
        if (trans && trans.data &&
trans.data.length > 0) {
          const getUsersItems =
getUsers(trans.data.map(item =>
item.userRecordKey))
          const users = await
Promise.all(getUsersItems)
          const transactionItems =
trans.data.map((value, index) => ({
            ...value,
            user: users[index].phoneNumber
          }))
          let pagenNumber = 1
          if (transactionItems.length >
10) {
            pagenNumber =
Math.ceil(transactionItems.length /
perPage)
          }
          setPages(pagenNumber)

setTransactions(transactionItems)
          setLoading(false)
        } else {
          setLoading(false)
        }
      } else {
        handleAlert('error', 'La fecha
inicio debe ser menor a la fecha de fin')
      }
    } catch (err) {
      handleAlert('error', err.message)
      setLoading(false)
    }
  };

  const handleChange = (event) => {
    const { value, name } = event.target
    switch (name) {
      case 'fromDate':
        setFromDate(value)
        callApi(value, toDate)
        break;
      case 'toDate':
        setToDate(value)
        callApi(fromDate, value)
        break;
      case 'searchedWord':
        setSearchedWord(value)
        break;
      default:
```

```
        handleAlert('error', `Input no
válido ${name}`)
    }
  }

  const handlePagination = (event, value)
=> {
    setSelectedPage(value)
  }

  const setMoneyFormat = (number) => {
    if (number != 'N/A') {
      return '$ ' +
number.toString().replace(/\B(?=(\d{3})+(?
!\d))/g, ",");
    } else {
      return number
    }
  }

  const AttachmentFile = () => {
    const inputRef = useRef(null);
    return (
      <div
className={classes.buttonWithCaption}>
        <input
          type="file"
          ref={inputRef}
          hidden
          accept=".txt"
          onChange={CargarErrores}
        />
        <Button

className={classes.defaultButton}
          onClick={event => {
            event.preventDefault();
            inputRef.current.click();
          }}
        >
          <img width="18" height="20.88"
src={iconUpload} alt="" />
          <span
className={classes.depositButtonText}>CARG
AR COMPROBANTES</span>
        </Button>
        <a href="#" onClick={(event) =>
{event.preventDefault(); abrirAyuda();}}
className={classes.captionForButton}>Ayuda
</a>
      </div>
    );
  };
  const abrirAyuda = () => {
    setAyudaUpload(!ayudaUpload);
  }
  const buildstatus = (status) => {
    if (status) {
      switch (status) {
        case 'PENDIENTE':
          return <Typography
className={classes.statusPending}><ul><li>
PENDIENTE</li></ul></Typography>
        case 'ENVIADO':
          return <Typography
className={classes.statusSending}><ul><li>
ENVIADO</li></ul></Typography>
        case 'REEMBOLSADO':
          return <Typography
className={classes.statusRefund}><ul><li>R
EEMBOLSADO</li></ul></Typography>
        case 'COMPLETADO':
          return <Typography
className={classes.statusCompleted}><ul><l
i>COMPLETADO</li></ul></Typography>
        default:
          return
<Typography><ul><li>{status}</li></ul></Ty
pography>
      }
    } else {
      return null
    }
  }

  const handleOpen = async (merchant,
isMerchant) => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  const saveSettings = async () => {
    let hasErros = false;
    if (!CheckValue("empty", RFC) ||
!CheckValue("rfc_moral", RFC)) {
      hasErros = true;
      handleAlert("error", 'Por favor
ingrese un RFC válido.');
    }
    if (!CheckValue("empty",
sourceAccount) || !CheckValue("integer",
sourceAccount)) {
      hasErros = true;
      handleAlert("error", 'La cuenta solo
debe contener valores numéricos.');
    }
    if (commission <= 0) {
      hasErros = true;
      handleAlert("error", 'La comisión
debe ser mayor a 0.');
    }
    if (description.length > 30) {
      hasErros = true;
      handleAlert("error", 'La descripción
debe ser menos de 30 caracteres');
    }
    if (!hasErros) {
      setLoading(true)
      try {
        let setup = {
          RFC: RFC,
          sourceAccount: sourceAccount,
          commission: commission,
          description: description
        }
        await API.put('setting-api',
`/settings/default_spei_config`, {
          body: {
            setup
          },
        });
        handleClose()
        handleAlert('success',
'Actualizado con Éxito!')
        setLoading(false)
      } catch (error) {
        setLoading(false)
        console.log('ERROR', error);
        handleAlert('error',
error.message)
      }
    }
  }

  const body = (
    <div style={modalStyle}
className={classes.paper}>
      <div
className={classes.mainContainerRegister}>
        <div className="SubContainer">
          <Typography
className={classes.title}>
            Configuraciones
          </Typography>
          <FormControl
className="inputForm col-12 col-sm-12
col-lg-12 col-xl-12">
            <InputLabel
className="labelInputStyle">RFC</InputLabe
l>
            <Input
              id="RFC"
              type="text"
              name="RFC"
              className="defaultInputText"
              value={RFC}
              onChange={(event) =>
setRFC(event.target.value.toUpperCase())}
            />
          </FormControl>
          <FormControl
className="inputForm col-12 col-sm-12
col-lg-12 col-xl-12">
            <InputLabel
className="labelInputStyle">Cuenta
Origen</InputLabel>
            <Input
              id="sourceAccount"
              type="text"
              name="sourceAccount"
              className="defaultInputText"
              value={sourceAccount}
              onChange={(event) =>
setSourceAccount(event.target.value)}
            />
          </FormControl>
          <FormControl
className="inputForm col-12 col-sm-12
col-lg-12 col-xl-12">
            <InputLabel
className="labelInputStyle">Comisión</Input
tLabel>
            <Input
              id="commission"
              type="number"
              name="commission"
              className="defaultInputText"
              value={commission}
              onChange={(event) =>
setCommission(event.target.value)}
            />
          </FormControl>
          <FormControl
className="inputForm col-12 col-sm-12
col-lg-12 col-xl-12">
            <InputLabel
className="labelInputStyle">Descripción</I
nputLabel>
            <Input
              id="description"
              type="text"
              name="description"
              className="defaultInputText"
              value={description}
              onChange={(event) =>
setDescription(event.target.value)}
            />
          </FormControl>
          <div
className={classes.modalButtons}>
            {!isLoading ?
              <>
                <Button
className={classes.buttonStyle}
onClick={saveSettings}>
                  <Typography
className={classes.buttonText}
>GUARDAR</Typography>
                </Button>
              </>
              :
              <>
                <Button
className={classes.buttonStyle}
disabled={true}>
                  <Typography
className={classes.buttonText}>GUARDANDO..
.</Typography>
                </Button>
              </>
            }
          </div>
        </div>
      </div>
    </div>
  );

  return (
    <div className={classes.root}>
      <Modal
        open={open}
        onClose={handleClose}

aria-labelledby="simple-modal-title"

aria-describedby="simple-modal-description
"
      >
        {body}
      </Modal>
      {isLoading ?
        <div className='gralinfo_form'>
          <CircularProgress
color="secondary" />
        </div>
        :
        <div
className={classes.container}>
          <div
className={classes.titleConteiner}>
```

```
          <div className="row col-12
col-sm-12 col-md-12 col-lg-12 col-xl-12">
            <div className="col-3
col-sm-3 col-md-3 col-lg-3 col-xl-3">
              <Typography
className={classes.title}>Transacciones
con SPEI</Typography>
            </div>
            <div
className={`${classes.titleFlex} col-9
col-sm-9 col-md-9 col-lg-9 col-xl-9
justify-content-end`}>
              <div
className={classes.buttonWithCaption}>
                <Button
className={classes.defaultButton}
                  onClick={event => {
getSpeiFiles(); }}>
                  <img width="18"
height="20.88" src={iconDownload} alt=""
/>
                  <span
className={classes.depositButtonText}>Desc
argar Movimientos</span>
                </Button>
                <span
className={classes.captionForButton}>Últim
o SPEI procesado:
{moment(lastSpeiProcessed).format('DD/MM/Y
YYY h:mm:ss a')}</span>

              </div>
              <AttachmentFile />
              <Button

className={classes.settingsButtonStyle}
                onClick={handleOpen}
              >
                <img width="22"
height="22" src={engine} alt="" />
                <span
className={classes.depositButtonText}>CONF
IGURACIONES</span>
              </Button>

            </div>
          </div>
        </div>
        <div
className={classes.transactionsConteiner}>
          <div className="row">
            <div className="col-lg-8
col-sm-12">
              <div className={`row
${classes.margin20}`}>
                <form
className={classes.formDate} noValidate>
                  <TextField
                    id="datetime-from"
                    name="fromDate"
                    value={fromDate}

onChange={handleChange}
                    label="Desde"

type="datetime-local"

defaultValue="2017-05-24T10:30"
                    InputLabelProps={{
                      shrink: true,
                    }}
                  />
                </form>
                <form
className={classes.formDate} noValidate>
                  <TextField
                    id="datetime-to"
                    name="toDate"
                    value={toDate}

onChange={handleChange}
                    label="Hasta"

type="datetime-local"

defaultValue="2017-05-24T10:30"
                    InputLabelProps={{
                      shrink: true,
                    }}
                  />
                </form>
              </div>
              <div
className={`${classes.titleFlex} col-lg-4
col-sm-12 justify-content-end`}>
                <DeafultSearch

className="outlinedSearchSmall"
                  label="Buscar por
usuario, orden"
                  name="searchedWord"
                  value={searchedWord}

handleChange={handleChange}
                />
              </div>
            </div>
            {transactions.length > 0 ?
<div className="mt-3 overflow-auto">
              <table className="table">
                <thead>
                  <tr>
                    <th>
                      <Typography
className={classes.tableHeaderText}>Usuari
o</Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderNumber}>Mont
o</Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderNumber}>Comi
sión</Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderText}>CLABE<
/Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderText}>No. de
Transacción</Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderCenter}>Fech
a y Hora (Hora Centro)</Typography>
                    </th>
                    <th>
                      <Typography
className={classes.tableHeaderCenter}>Esta
do</Typography>
                    </th>
                  </tr>
                </thead>
                <tbody>

{transactions.slice(((selectedPage - 1) *
perPage), (((selectedPage - 1) * perPage)
+ perPage)).map((transaction) => (
                    (searchedWord.length <
3 ||
JSON.stringify(transaction).toUpperCase().
indexOf(searchedWord.toUpperCase()) !==
-1) && <tr key={transaction.id}>
                      <td>
                        <Typography
className={classes.tableRowText}>{transact
ion.user}</Typography>
                      </td>
                      <td
className={classes.backgrounOddColumn}>
                        <Typography
className={classes.tableRowNumber}>{setMon
eyFormat(transaction.amount.toFixed(2))}</
Typography>
                      </td>
                      <td>
                        <Typography
className={classes.tableRowNumber}>{setMon
eyFormat((transaction.subAmount -
transaction.amount).toFixed(2))}</Typograp
hy>
                      </td>
                      <td
className={classes.backgrounOddColumn}>
                        <Typography
className={classes.tableRowText}>{transact
ion.clabe.includes('CLABE') ?
transaction.clabe :
'DEBITO*******'}</Typography>
                      </td>
                      <td>
                        <Typography
className={classes.tableRowText}>{transact
ion.accountRecordKey}</Typography>
                      </td>
                      <td
className={classes.backgrounOddColumn}>
                        <Typography
className={classes.tableRowCenter}>{moment
(transaction.createdAt).format('DD/MM/YYYY
h:mm:ss a')}</Typography>
                      </td>
                      <td>
                        <Typography
className={classes.tableRowCenter}>{builds
tatus(transaction.status)}</Typography>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
              <div
className={classes.paginationTAB}>
                <Pagination count={pages}
shape="rounded"
onChange={handlePagination} />
              </div>
            </div>
              :
              <div
className={classes.errorContainer}>
                <img src={noMovements}
alt=""></img>
                <Typography
className={classes.errorText} >No se
encontraron transacciones</Typography>
                <Typography
className={classes.errorText2} >Por favor
selecciona otro rango de
fechas</Typography>
              </div>
            }
          </div>
        </div>
      }
      {ayudaUpload && <AyudaModal
open={ayudaUpload} cancel={abrirAyuda} />}
    </div>
  );
}

export default withSnackbar(Spei)
```

Creación del crontab que genera los documentos SPEI cada 30 minutos

```yaml
buildSPEIfiles:
  handler: buildSpeiFiles.build
  events:
    - schedule: cron(0/30 * ? * MON-FRI *)
    - http:
        path: speis/batch
        method: any
        authorizer:
          type: aws_iam
        cors:
          origin: '*'
          headers:
            - Content-Type
            - X-Amz-Date
            - Authorization
            - X-Api-Key
            - X-Amz-Security-Token
            - X-Amz-User-Agent
```

Creación de la tabla para guardar analiticos de los archivos SPEI (por cuestiones de privacidad no se permite mostrar el nombre completo ni resources ni enviroment con la que es llamada)

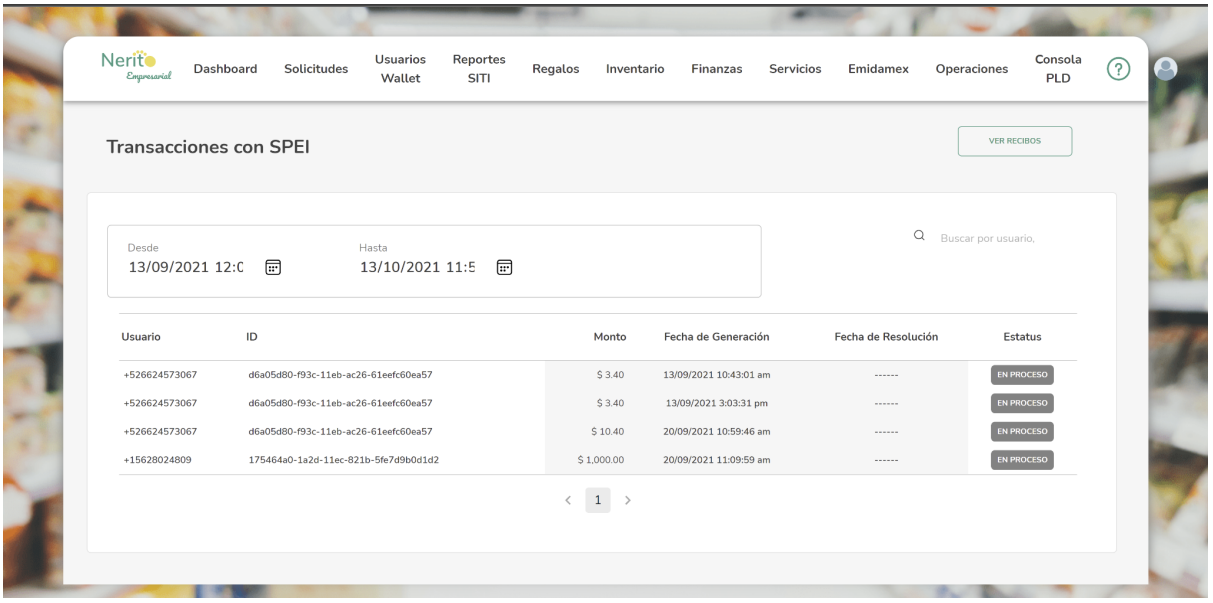
```yaml
173     ReceiptSpei:
174       Type: AWS::DynamoDB::Table
175       DeletionPolicy: Retain
176       Properties:
177         AttributeDefinitions:
178           - AttributeName: idOperation
179             AttributeType: S
180           - AttributeName: createdAt
181             AttributeType: N
182         KeySchema:
183           - AttributeName: idOperation
184             KeyType: HASH
185           - AttributeName: createdAt
186             KeyType: RANGE
187         ProvisionedThroughput:
188           ReadCapacityUnits: 2
189           WriteCapacityUnits: 1
190         StreamSpecification:
191           StreamViewType: NEW_IMAGE
192         TableName: receiptsspeis-[redacted]
```

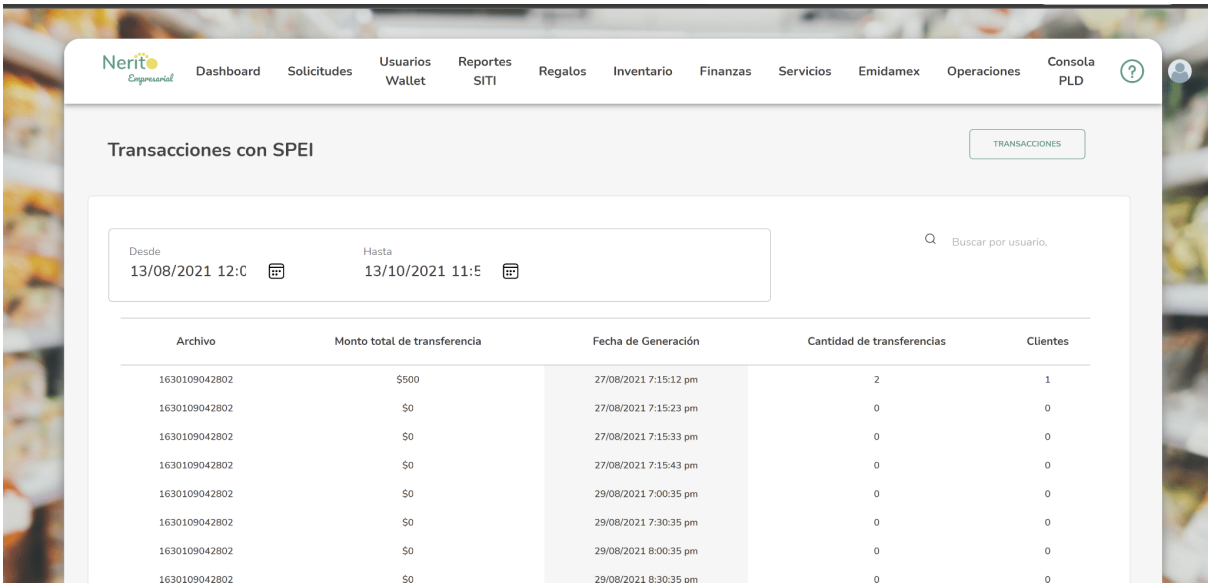
Codificación de variables para las tuplas y metodos get y put

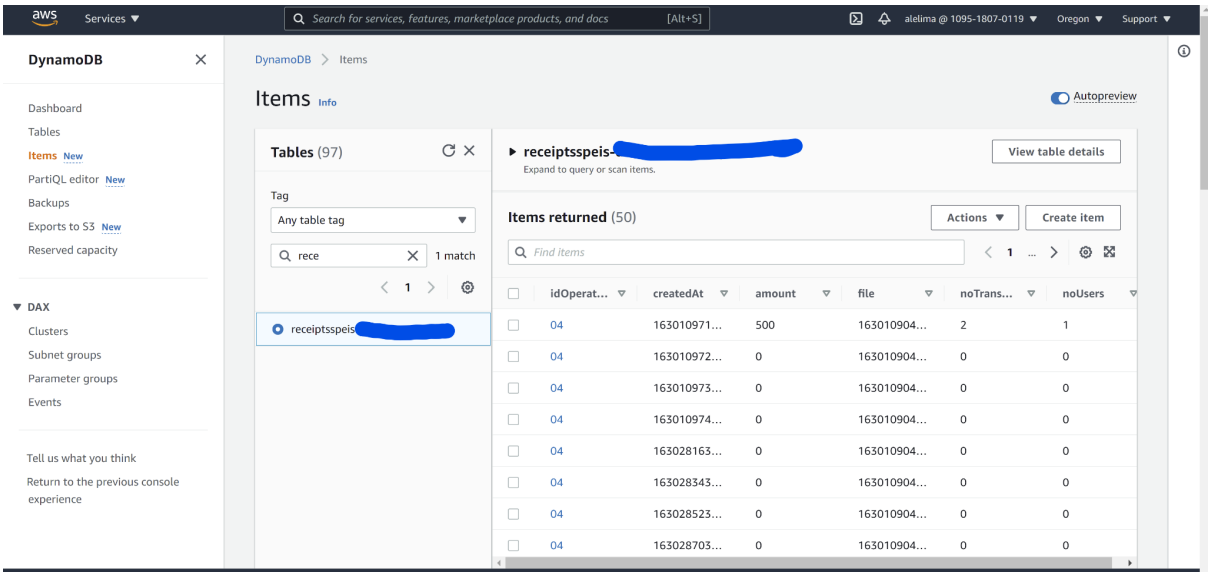
```
//ARCHIVO APP.JS CON METODO GET
app.get(`${path}/receipts`, async function (req, res) {
  try {
    const today = moment.utc(moment.now()).valueOf();
    const since = req.query['since'] && parseInt(req.query['since'])
|| 0;
    const to = req.query['to'] && parseInt(req.query['to']) || today;
    const query = {
      TableName: process.env.STORAGE_RECEIPTSPEIS,
      KeyConditionExpression: 'idOperation = :op AND createdAt
BETWEEN :since AND :to',
      ExpressionAttributeValues: {
        ":op":'04',
        ":since": since,
        ":to": to,
      },
    };
    const data = await dynamodb.query(query).promise();
    res.json({ success: true, data });
  } catch (error) {
    console.error('error obteniendo lista de speis', error);
    res.statusCode = 500;
    res.json({ success: false, error });
  }
})

app.get(`${path}/:userRecordKey`, async function (req, res) {
  try {
    const userRecordKey = req.params['userRecordKey'];
    const today = moment.utc(moment.now()).valueOf();
    const since = req.query['since'] && parseInt(req.query['since'])
|| 0;
    const to = req.query['to'] && parseInt(req.query['to']) || today;
    const query = {
      TableName: process.env.STORAGE_SPEIS,
      KeyConditionExpression: 'operacion = :op AND createdAt BETWEEN
:since AND :to',
      FilterExpression: 'userRecordKey = :urk',
      ExpressionAttributeValues: {
        ":since": since,
        ":to": to,
        ":op": operacion.SPEI_04,
        ":urk": userRecordKey
      },
    };
    querySpei(query, res);
  } catch (error) {
    console.error('error obteniendo lista de speis', error);
    res.statusCode = 500;
    res.json({ success: false, error });
  }
})


//ARCHIVO BUILDSPEIFILES DONDE SE REALIZA EL .PUT
var total = 0;
var iteraciones = 0;
var userst = [];

if(operacion ==='04'){
    console.log("settings.setup.lastSpeiProcessed aca =
"+settings.setup.lastSpeiProcessed);
    Array.prototype.unicos = function () {
      return this.filter((valor, indice) => {
        return this.indexOf(valor) === indice;
      });
    }
    const userst = users.unicos();
    const today = moment.utc(moment.now()).valueOf();
    const item ={
      createdAt: today,
      idOperation: '04',
      file: settings.setup.lastSpeiProcessed,
      amount: total,
      noTransactions: iteraciones,
      noUsers: userst.length,
    }
    const queryParams = {
      TableName: process.env.STORAGE_RECEIPTSPEIS,
      Item: item,
```

```
};
const response = await dynamodb.put(queryParams).promise();
```

```
}
```

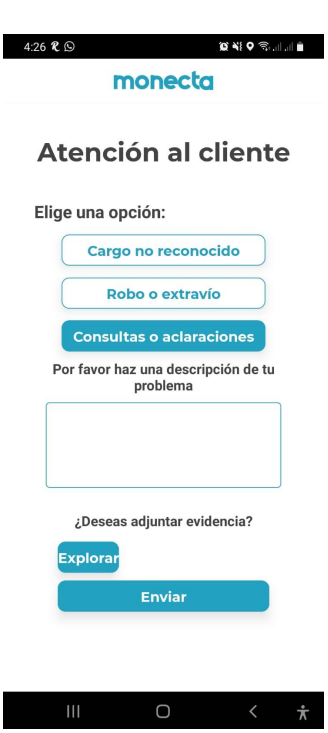


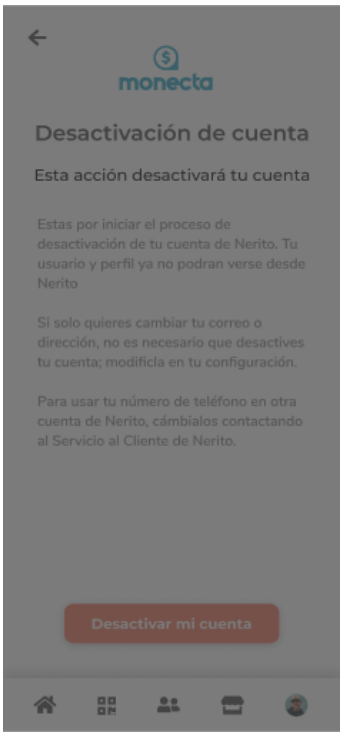




Emma:
Creación de cuestionario PLD (prevención de lavado de dinero)
Atención al cliente
Desactivación de cuenta

1. Tener el repositorio instalado
2. Entender en qué parte del flujo se requiere la pantalla y se ve diseño en Figma (app web para crear y compartir diseños)
3. Buscar componentes disponibles (se reciclan para ahorrar espacio y optimizar el código). Estos son los botones, selects, checkboxes. Se crean loc componentes que no existen en caso de hacer falta.
4. Programar la pantalla y luego probar que funcione.
5. Conectar con la base de datos y probar que se estén guardando los datos correctamente.
6. Mandar solicitud para que se aprueben los cambios.





En la primera parte del código se importan los componentes con sus rutas

```
import React, { useState } from 'react';
import {
  View,
  Text,
  StyleSheet,
  Image,
  TouchableOpacity,
  ScrollView
} from 'react-native';
import theme from '../../styles/theme';
import Select from '../../components/Select'
import PrimaryInput from '../../components/PrimaryInput'
import PrimaryButton from '../../components/PrimaryButton';
import SecondaryButton from '../../components/SecondaryButton';
import { CheckBox } from 'react-native-elements';
import saveQuestionnaire from '../../lib/profile/saveQuestionnaire';
import { connect } from 'react-redux';
import LoginActions from '../../redux/Login';
import MyProfileActions from '../../redux/MyProfile';
```

Se declaran las variables; se usan estados o states.

```
const [professionSelected, setProfessionSelected] = useState('');
const [ocupationSelected, setOcupationSelected] = useState('');
```

Se ponen las funciones

```
const setSelected = (key, index) => {
  switch (key) {
    case 'sources':
      setSourceSelected(sources[index]);
      setSourceError({ showError: false, error: '' });
      break;
    case 'professions':
      setProfessionSelected(professions[index].value);
      setProfessionError({ showError: false, error: '' });
      break;
    case 'ocupations':
      setOcupationSelected(ocupations[index].value);
      setOcupationError({ showError: false, error: '' });
      break;
    default:
      break;
  }
}
```

En el return se mete la parte visual de la pantalla y la llamada a las funciones

```
return (
  <ScrollView style={styles.root}>
    <View style={styles.title}>
      <Text style={styles.verifyMessage}>Cuestionario PLD</Text>
    </View>
    <View style={styles.infoContainer}>
      <Text style={styles.label}>
        Monto mensual a transaccionar
      </Text>
      <View style={styles.checkboxAndTextContainer}>
        <CheckBox
          containerStyle={styles.checkboxContainer}
          checkedIcon='dot-circle-o'
          uncheckedIcon='circle-o'
          checked={monthlyAmount === '$0 - $5,000'}
          onPress={() => setAmount('$0 - $5,000')}
          textStyle={styles.label}
        />
```

Siempre en la última parte del código se hace una hoja de estilos, que son usados en los componentes y textos para alinearlos y darles un formato.

```
const styles = StyleSheet.create({
  root: {
    backgroundColor: '#ffffff',
    flex: 1,
  },
  checkboxContainer: {
    backgroundColor: theme.palette.white,
    borderWidth: 0,
    padding: 0,
  },
  checkboxAndTextContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    paddingRight: 25,
  },
  image: {
    alignSelf: 'center',
    marginTop: 20,
  },
  title: {
    margin: 30,
  },
```

Llamadas al backend/ base de datos: el encargado de las bases de datos da el link en el que se hace la conexión y comprueba que sí esté llegando la información.

```
const formData = new FormData();
formData.append('attachment', attachments);
formData.append('contactName', contactName);
formData.append('email', email);
formData.append('phoneNumber', phoneNumber);
formData.append('description', description);
formData.append('userID', userID);
const response = await  fetch("https://hooks.zapier.com/hooks/catch/8872951/o03301k/", {
method: 'POST',
  body: formData,
}).then(response => response.json())
.then(data => {
  console.log('SUCCESS', data);
  props.navigation.navigate('SuccessScreen', {
    from: 'CustomerService',
    messageTitle: '¡Listo!'
  });
})
.catch((error) => {
  console.log('ERR', error);
});
```

Comúnmente te lleva a la pantalla de éxito cuando todo se envía correctamente y en el caso contrario aparece un mensaje (toast) con el error.

Las llamada al back y uso de las APIs se puede hacer directamente o dividir en 3 etapas Cuando se divide se tienen que mandar un conjunto de datos al código de la conexión para que los mande a la API del back y los publique.

```
/**
 * SaveProfile process for Cognito
 *
 * @param {string} monthlyAmount
 * @param {boolean} pep
 * @param {string} source
 * @param {string} profession
 * @param {string} ocupation
 * @returns {object} with success, message, errorType
 */
const SaveProfile = async (userRecordKey, data) => {
    const {
        monthlyAmount,
        isPep,
        source,
        profession,
        ocupation,
    } = data;
```

```
try {
    const result = await API.post(API_PLD, `/pld/user`, {
        body: {
            monthlyAmount: data.monthlyAmount,
            ocupation: data.ocupation,
            profession: data.profession,
            isPEP: data.isPep,
            resourceOrigin: data.source,
    userRecordKey: userRecordKey,
        },
    })

    return {
        success: true,
        user: result.data[0],
    }
```

Conclusiones

Alejandro: Dentro del proceso de transacciones financieras en una fintech, se deben de tomar en cuenta la generación de archivos SPEI, adicional a ello, debemos de garantizar el conocimiento de dichas transacciones y la mejor manera es haciéndolo en vista, estableciendo consolas o dashboards que muestren cómo se comporta la población que utiliza la misma. El hecho de generar una vista web, puede implicar la modificación o creación de nuevas tablas en las bases de datos, dependiendo la infraestructura tecnológica con la que se cuente, en este caso aws, se debe de seguir la documentación y con ayuda de ingenieros más calificados, consolidar los cambios y evaluar la solución con la que se atacará el problema. Estoy satisfecho del aprendizaje en estas tareas, me han permitido conocer más sobre aws console y de servicios como figma para el diseño gráfico así como poder combinar el aprendizaje adquirido en la materia de desarrollo web para poder establecer métodos de programación para llegar al resultado esperado.

Emma: Se cumplieron con los objetivos declarados en el practicum y los establecidos por la empresa, se mostraron conocimientos de las asignaturas de software mencionadas en los objetivos. Se respetó el tiempo del calendario y la metodología descrita. Se aumentaron los conocimientos relativos a la programación móvil, creación de funciones, las APIs y su implementación.

Calendario

MONECTA aplicación regulada	**31 days**	**Mon 8/23/21 8:**	**Mon 10/4/21 5:**		**Late**	**64%**
Desarrollo de pantallas	**31 days**	**Mon 8/23/21 8:**	**Mon 10/4/21 5:**		**Late**	**64%**
Perfil	**3 days**	**Wed 9/1/21 8:0**	**Fri 9/3/21 5:00 I**		**Late**	**95%**
Cuestionario PLD	3 days	Wed 9/1/21 8:0	Fri 9/3/21 5:00 I	Emma Alvarez	Complete	100%
Opciones mi cuenta	**2 days**	**Mon 8/23/21 8:**	**Tue 8/24/21 5:0**		**Complete**	**100%**
Atención a cliente	2 days	Wed 9/1/21 8:0	Thu 9/2/21 5:00	Emma Alvarez	Complete	100%
Desactivar cuenta	2 days	Wed 9/1/21 8:0	Thu 9/2/21 5:00	Emma Alvarez	Complete	100%

Consola de administración - SPEI	21d	20/09/21	18/10/21		44%	En progreso
Backend	16d	27/09/21	18/10/21		43%	Completo
Backend - Realizacion de un crontab de generacion de documentos de transacciones SPEI cada 30 min	1d	27/09/21	27/09/21	AL Alejandro Lima	100%	Completo
Backend - Almacenar información de layouts SPEI	4d	28/09/21	01/10/21	AL Alejandro Lima	100%	Completo
Backend - Crear end point para reproceso de transacciones y reflejar los analiticos en la tabla creada	4d	04/10/21	07/10/21	AL Alejandro Lima	100%	Completo
Backend - Actualizar status de transacciones con archivo de respuesta BEM	12d	01/10/21	18/10/21	AL Alejandro Lima	0%	Sin iniciar
Frontend	16d	20/09/21	11/10/21		56%	
Front end - Crear pantalla de resumen de transacciones	2d	04/10/21	05/10/21	NC Nancy Colín	0%	Sin iniciar
Front end - Crear pantalla para reproceso de transacciones	2d	08/10/21	11/10/21	NC Nancy Colín	0%	Sin iniciar
Front end - Crear rediseño de consola SPEI en figma	2d	20/09/21	21/09/21	AL Alejandro Lima	100%	Completo
Front end - Consola proceso SPEI	3d	22/09/21	24/09/21	AL Alejandro Lima	100%	Completo
Configurar notificaciones a usuario por excepciones, reprocesos	2d	08/10/21	11/10/21	FS Fernando Sanchez	0%	Sin iniciar

En el siguiente cuadro, se marcan con color amarillo las etapas sobre las que se trabajo presentadas en el protocolo, en este caso en las primeras 4 semanas no hemos tenido que llevar a cabo las tareas de respuestas al regulador (CNBV), ya que no se nos ha solicitado. Sin embargo se llevaron a cabo las tareas correspondientes a las etapas:

- Entendimiento e instalación de herramientas
- Testeo de las aplicaciones

- Resolución de bugs, mejoras y creación de nuevas funcionalidades

Calendario de actividades para el desarrollo de la aplicación Nerito en las primeras 4 semanas

		Semanas												
Etapa(s)	Actividades	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1													
	2													
	3													
	4													
	5													
2	6													
	7													
	8													
3	9													
	10													
4	11													
	12													
	13													

Referencias

- Programming with DynamoDB and the AWS SDKs - Amazon DynamoDB. (s. f.). AWS console. Recuperado 1 de octubre de 2021, de https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Programming.html

- Programación de expresiones con rate o cron - AWS Lambda. (s. f.). AWS console. Recuperado 1 de octubre de 2021, de https://docs.aws.amazon.com/es_es/lambda/latest/dg/services-cloudwatchevents-expressions.html

- Tutorial de GitLab: instalación y primeros pasos. (2020, 21 octubre). IONOS Digitalguide. Recuperado 14 de octubre de 2021, de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/tutorial-de-gitlab/

- Next U. (2017, 30 agosto). ¿QUÉ ES Y CÓMO FUNCIONA REACT.JS? NextU LATAM. Recuperado 14 de octubre de 2021, de https://www.nextu.com/blog/que-es-y-como-funciona-react-js/

- ASP 9. (s. f.). ASP. Recuperado 14 de octubre de 2021, de https://www.uv.es/jac/guia/asp/asp9.htm

- ¿Qué es una API? (s. f.). RedHat. Recuperado 14 de octubre de 2021, de https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

- Barrera, A. (2019, 10 noviembre). JSON: ¿Qué es y para qué sirve? NextU LATAM. Recuperado 14 de octubre de 2021, de https://www.nextu.com/blog/que-es-json/