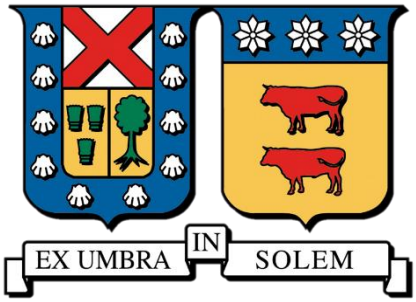


Tarea 3: Cassandra

Integrantes/Rol/Paralelo:

Benjamín Nicolas Daza Jiménez/202173574-7/201

Sebastián Andrés Von Kunowsky Lepe/202173560-7/201

Curso: Bases de datos avanzadas

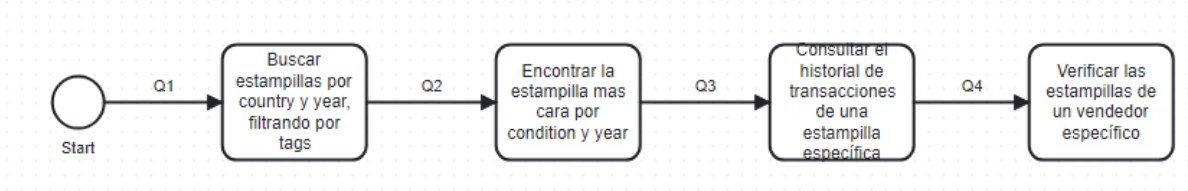
Profesor: José Luis Martí Lara

Ayudante: Daniela Sánchez Nizza

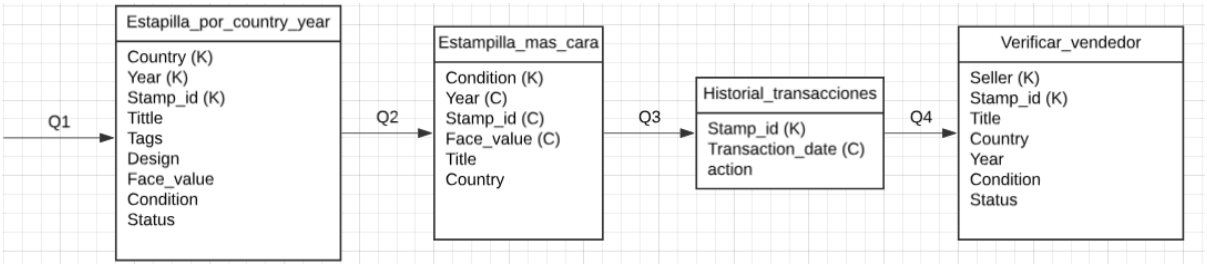
En el archivo “Tarea3.js” se encuentra la implementación de las respuestas (querys) de cada una de las preguntas, las cuales se puede acceder a través del menu por terminal.

1) (Item2) Modelos:

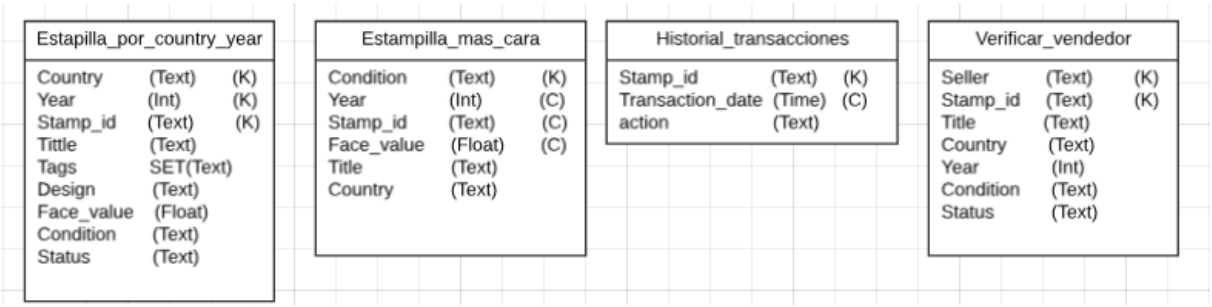
Navegación de consultas:



Modelo de datos lógico:



Modelo de datos físicos:



2) (Item2) Buscar estampillas por country y year. Filtrando por tags:

```
async function buscar_estampilla(){
  //<-- Variables -->
  const country = await preguntar("Ingrese country: ");
  const year = await preguntar("Ingrese year: ");
  const tag = await preguntar("Ingrese tag: ");
  const query = `
    SELECT * FROM stamps
    WHERE country = ? AND year = ? AND tags CONTAINS ? ALLOW FILTERING;
  `;
  try {
    const result = await client.execute(query, [country, year, tag], { prepare: true });
    result.rows.forEach(row => {
      // Formatear los valores en time_value
      const timeValueFormatted = row.time_value.map(tuple => ({
        date: tuple.get(0),
        value: tuple.get(1)
      }));

      // Formatear los valores en transaction_history
      const transactionHistoryFormatted = row.transaction_history.map(tuple => ({
        date: tuple.get(0),
        action: tuple.get(1)
      }));

      // Crear un nuevo objeto sin el campo stamp_id
      const { stamp_id, ...dataWithoutId } = row;

      console.log(`Estampilla encontrada - ID: ${stamp_id.toString()}`);
      console.log("Datos:", {
        ...dataWithoutId,
        time_value: timeValueFormatted,
        transaction_history: transactionHistoryFormatted
      });
    });
  } catch (error) {
    console.error('Error al ejecutar la consulta:', error);
  }
}
```

La función busca por country y year, filtrando por tags. Primero pide que se ingrese el country, year y un tag, luego se ejecuta una query que selecciona la estampilla que coincidan con los filtros y muestra por pantalla toda la información de la plantilla.

```
Ingrese country: Chile
Ingrese year: 2010
Ingrese tag: historia
Estampilla encontrada - ID: baecaf79-2a65-40e1-b5d2-66914b5233e5
Datos: {
  condition: 'dañado',
  country: 'Chile',
  design: 'Imagen conmemorativa',
  face_value: 10,
  seller: 'Ana López',
  series: 'Bicentenario',
  status: 'disponible',
  tags: [ 'bicentenario', 'historia' ],
  time_value: [ { date: '2024-11-09', value: 10 } ],
  title: 'Estampilla Bicentenario',
  transaction_history: [ { date: '2024-11-09', action: 'venta' } ],
  year: 2010
```

3) (Item2) Encontrar la estampilla más cara por condición y año:

```
async function mas_cara(){
  const condition = await preguntar("Ingrese condition: ");
  const year = await preguntar("Ingrese year: ");
  const query = `
    SELECT * FROM stamps
    WHERE condition = ? AND year = ? ALLOW FILTERING;
  `;
  try {
    const result = await client.execute(query, [condition, year], { prepare: true });
    if (result.rowLength > 0) {
      const mostExpensiveStamp = result.rows.reduce((max, row) => {
        return row.face_value > (max.face_value || 0) ? row : max;
      }, {});
      console.log('La estampilla más cara:', mostExpensiveStamp);
    } else {
      console.log('No se encontraron estampillas con los criterios dados.');
    }
  } catch (error) {
    console.error('Error al ejecutar la consulta:', error);
  }
}
```

La función busca la estampilla más cara dada una condición y un año. Primero se le pide al usuario que ingrese una condition y luego se pide que ingrese un year. Luego se ejecuta una query que selecciona las estampillas que cumplen con la condición y el año, luego se busca la estampilla más cara.

```
Ingrese su opción: 2
Ingrese condition: nuevo
Ingrese year: 2020
La estampilla más cara - ID: adfc3144-61da-43dc-8c27-1e12c531fc88, Datos: {
  condition: 'nuevo',
  country: 'Argentina',
  design: 'Retrato de figuras históricas argentinas',
  face_value: 7.5,
  seller: 'Carlos Domínguez',
  series: 'Bicentenario',
  status: 'disponible',
  tags: [ 'Argentina', 'bicentenario', 'historia' ],
  time_value: [ Tuple { elements: [Array], length: 2 } ],
  title: 'Estampilla del Bicentenario de Argentina',
  transaction_history: [ Tuple { elements: [Array], length: 2 } ],
  year: 2020
}
```

4) (Item2) Consultar el historial de transacciones de una estampilla específica:

```
async function historial() {
  const stampId  = await preguntar("Ingrese id: ");
  const query = `
    SELECT transaction_history FROM stamps
    WHERE stamp_id = ?;
  `;
  try {
    const result = await client.execute(query, [stampId], { prepare: true });
    if (result.rowLength > 0) {
      const transactionHistory = result.rows[0].transaction_history;
      console.log('Historial de transacciones:', transactionHistory);
    } else {
      console.log('No se encontró la estampilla con el ID proporcionado.');
    }
  } catch (error) {
    console.error('Error al ejecutar la consulta:', error);
  }
}
```

Esta función consulta el historial de transacción de una estampilla. Primero se le pide al usuario ingresar el id de la estampilla, se realiza una query para buscar la estampilla con ese id y luego se muestra por consola el historial de transacción por pantalla.

```
Ingrese su opción: 3
Ingrese id: 1aabd15f-a959-4b2a-9d1a-bf095df55734
Historial de transacciones: [ Tuple { elements: [ '2024-11-09', 'venta' ], length: 2 } ]
```

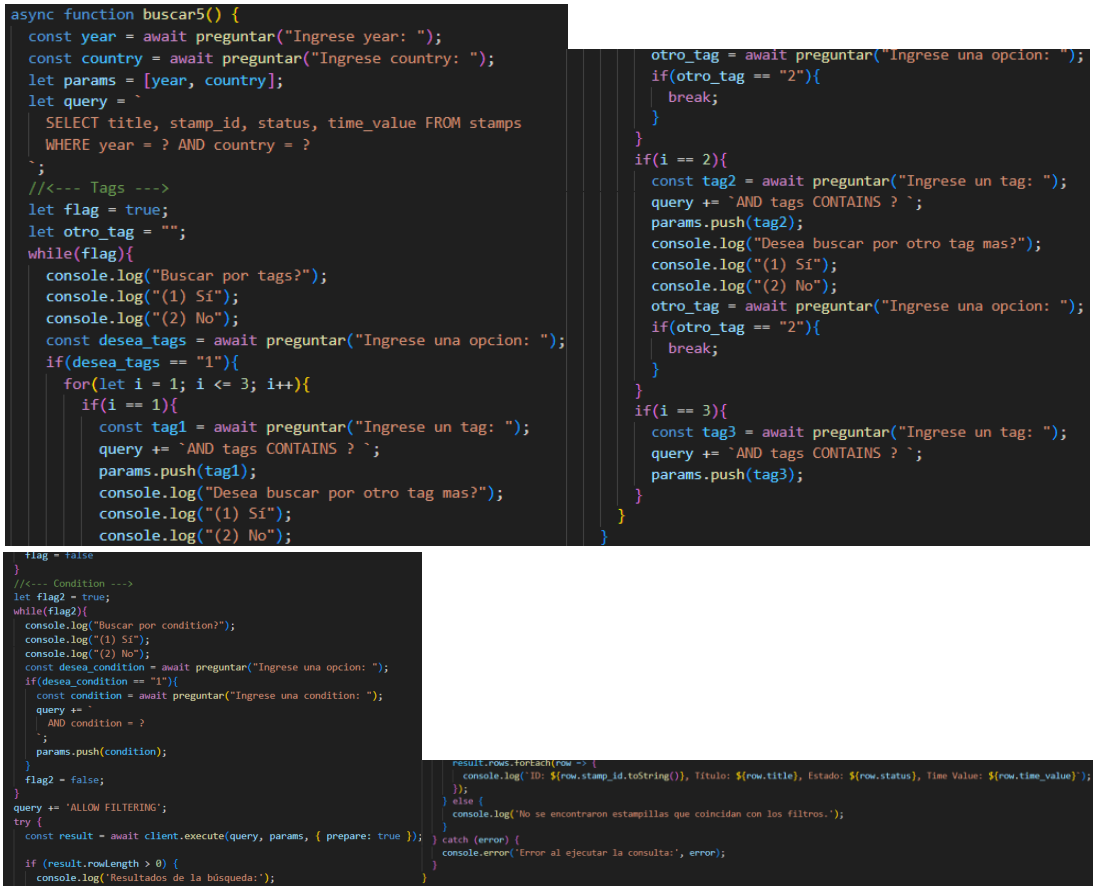
5) (Item2) Verificar las estampillas de un vendedor específico:

```
async function verificar() {
  const seller  = await preguntar("Ingrese seller: ");
  const query = `
    SELECT * FROM stamps
    WHERE seller = ?
    ALLOW FILTERING
  `;
  try {
    const result = await client.execute(query, [seller], { prepare: true });
    if (result.rowLength > 0) {
      console.log(`Estampillas de ${seller}:`);
      result.rows.forEach(row => {
        console.log(`ID: ${row.stamp_id.toString()}, Título: ${row.title}, Año: ${row.year}, Valor: ${row.face_value}`);
      });
    } else {
      console.log('No se encontraron estampillas para este vendedor.');
    }
  } catch (error) {
    console.error('Error al ejecutar la consulta:', error);
  }
}
```

La función muestra por pantalla todas las estampillas de un vendedor, primero se le pide al usuario que ingrese el vendedor, luego se realiza una query para buscar todas las estampillas de ese vendedor y luego se muestra por pantalla la información de cada estampilla.

```
Ingrese su opción: 4
Ingrese seller: Luc Dupont
Estampillas de Luc Dupont:
ID: 99064707-8645-42d8-a8b6-0077d3b79fd9, Título: Estampilla de la Unión Europea, Año: 2004, Valor: 2
ID: 92693862-8bb9-4b84-b893-86dcbbedae3e, Título: Estampilla de la Reina Isabel II, Año: 1953, Valor: 10
```

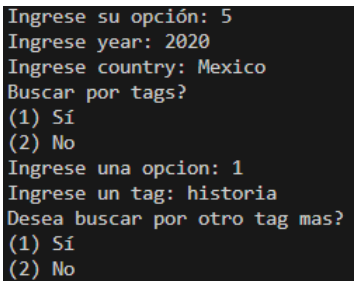
6) (Item3) Buscar por anio y pais. Opcionalmente por Tag y condition:



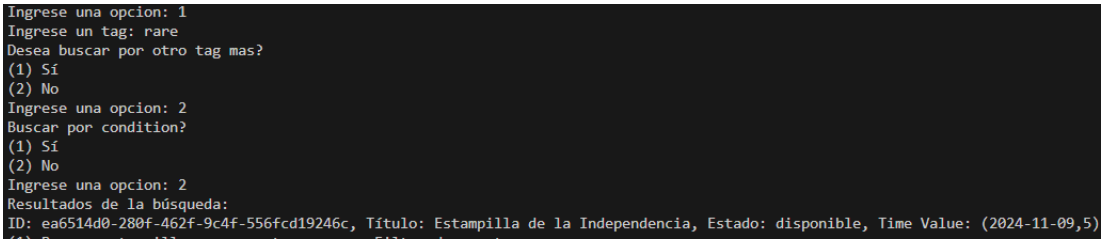
```
async function buscar5() {
  const year = await preguntar("Ingrese year: ");
  const country = await preguntar("Ingrese country: ");
  let params = [year, country];
  let query = `
    SELECT title, stamp_id, status, time_value FROM stamps
    WHERE year = ? AND country = ?
  `;
  //<--- Tags --->
  let flag = true;
  let otro_tag = "";
  while(flag){
    console.log("Buscar por tags?");
    console.log("(1) Sí");
    console.log("(2) No");
    const desea_tags = await preguntar("Ingrese una opcion: ");
    if(desea_tags == "1"){
      for(let i = 1; i <= 3; i++){
        if(i == 1){
          const tag1 = await preguntar("Ingrese un tag: ");
          query += `AND tags CONTAINS ? `;
          params.push(tag1);
          console.log("Desea buscar por otro tag mas?");
          console.log("(1) Sí");
          console.log("(2) No");
          otro_tag = await preguntar("Ingrese una opcion: ");
          if(otro_tag == "2"){
            break;
          }
        }
        if(i == 2){
          const tag2 = await preguntar("Ingrese un tag: ");
          query += `AND tags CONTAINS ? `;
          params.push(tag2);
          console.log("Desea buscar por otro tag mas?");
          console.log("(1) Sí");
          console.log("(2) No");
          otro_tag = await preguntar("Ingrese una opcion: ");
          if(otro_tag == "2"){
            break;
          }
        }
        if(i == 3){
          const tag3 = await preguntar("Ingrese un tag: ");
          query += `AND tags CONTAINS ? `;
          params.push(tag3);
        }
      }
    }
    flag = false
  }
  //<--- Condition --->
  let flag2 = true;
  while(flag2){
    console.log("Buscar por condition?");
    console.log("(1) Sí");
    console.log("(2) No");
    const desea_condition = await preguntar("Ingrese una opcion: ");
    if(desea_condition == "1"){
      const condition = await preguntar("Ingrese una condition: ");
      query += `
        AND condition = ?
      `;
      params.push(condition);
    }
    flag2 = false;
  }
  query += 'ALLOW FILTERING';
  try {
    const result = await client.execute(query, params, { prepare: true });

    if (result.rowLength > 0) {
      console.log('Resultados de la búsqueda:');
      result.rows.forEach(row => {
        console.log(`ID: ${row.stamp_id.toString()}, Título: ${row.title}, Estado: ${row.status}, Time Value: ${row.time_value}`);
      });
    } else {
      console.log('No se encontraron estampillas que coincidan con los filtros.');
    }
  } catch (error) {
    console.error('Error al ejecutar la consulta:', error);
  }
}
```

La función busca por año y país una estampilla, adicionalmente se puede buscar por tags (máximo tres) y por condición. Primero se le pregunta al usuario el año y pais de la estampilla, luego se pregunta si quiere buscar por algún tag (se le preguntará cada vez que agregue uno si quiere agregar otro, hasta un máximo de tres tags), posteriormente se le pregunta al usuario si quiere buscar por condición. Se realiza la query que busca con los datos dados por el usuario (la query se modifica a medida que el usuario decide agregar tags o condiciones) y se muestra por pantalla los datos de la estampilla (los solicitados).

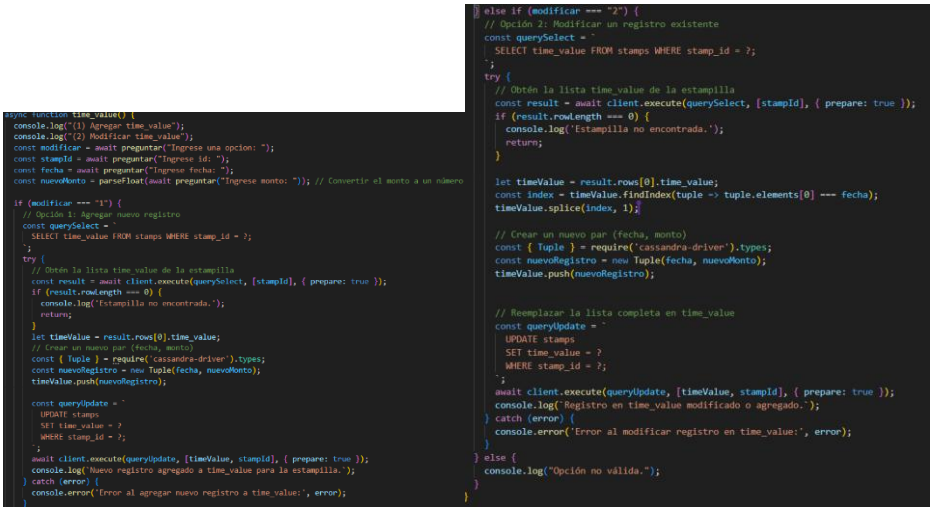


```
Ingrese su opción: 5
Ingrese year: 2020
Ingrese country: Mexico
Buscar por tags?
(1) Sí
(2) No
Ingrese una opcion: 1
Ingrese un tag: historia
Desea buscar por otro tag mas?
(1) Sí
(2) No
```



```
Ingrese una opcion: 1
Ingrese un tag: rare
Desea buscar por otro tag mas?
(1) Sí
(2) No
Ingrese una opcion: 2
Buscar por condition?
(1) Sí
(2) No
Ingrese una opcion: 2
Resultados de la búsqueda:
ID: ea6514d0-280f-462f-9c4f-556fcd19246c, Título: Estampilla de la Independencia, Estado: disponible, Time Value: (2024-11-09,5)
```

7) (Item4) Actualizar registro time_value:



```
async function time_value() {
  console.log("(1) Agregar time_value");
  console.log("(2) Modificar time_value");
  const modificar = await preguntar("Ingrese una opcion: ");
  const stampId = await preguntar("Ingrese id: ");
  const fecha = await preguntar("Ingrese fecha: ");
  const nuevoMonto = parseFloat(await preguntar("Ingrese monto: ")); // Convertir el monto a un número

  if (modificar === "1") {
    // Opción 1: Agregar nuevo registro
    const querySelect = `
      SELECT time_value FROM stamps WHERE stamp_id = ?;
    `;
    try {
      // Obtén la lista time_value de la estampilla
      const result = await client.execute(querySelect, [stampId], { prepare: true });
      if (result.rowLength === 0) {
        console.log('Estampilla no encontrada.');
        return;
      }
      let timeValue = result.rows[0].time_value;
      // Crear un nuevo par (fecha, monto)
      const { Tuple } = require('cassandra-driver').types;
      const nuevoRegistro = new Tuple(fecha, nuevoMonto);
      timeValue.push(nuevoRegistro);

      const queryUpdate = `
        UPDATE stamps
        SET time_value = ?
        WHERE stamp_id = ?;
      `;
      await client.execute(queryUpdate, [timeValue, stampId], { prepare: true });
      console.log('Nuevo registro agregado a time_value para la estampilla.');
    } catch (error) {
      console.error('Error al agregar nuevo registro a time_value:', error);
    }
  } else if (modificar === "2") {
    // Opción 2: Modificar un registro existente
    const querySelect = `
      SELECT time_value FROM stamps WHERE stamp_id = ?;
    `;
    try {
      // Obtén la lista time_value de la estampilla
      const result = await client.execute(querySelect, [stampId], { prepare: true });
      if (result.rowLength === 0) {
        console.log('Estampilla no encontrada.');
        return;
      }

      let timeValue = result.rows[0].time_value;
      const index = timeValue.findIndex(tuple => tuple.elements[0] === fecha);
      timeValue.splice(index, 1);

      // Crear un nuevo par (fecha, monto)
      const { Tuple } = require('cassandra-driver').types;
      const nuevoRegistro = new Tuple(fecha, nuevoMonto);
      timeValue.push(nuevoRegistro);


      // Reemplazar la lista completa en time_value
      const queryUpdate = `
        UPDATE stamps
        SET time_value = ?
        WHERE stamp_id = ?;
      `;
      await client.execute(queryUpdate, [timeValue, stampId], { prepare: true });
      console.log('Registro en time_value modificado o agregado.');
    } catch (error) {
      console.error('Error al modificar registro en time_value:', error);
    }
  } else {
    console.log("Opción no válida.");
  }
}
```

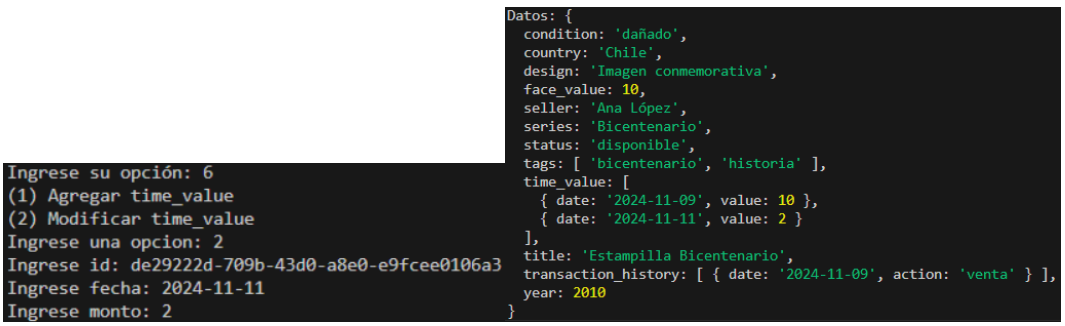
La funcion permite agregar o modificar un time_value de una estampilla, primero se le pide al usuario que quiere hacer (agregar o modificar), luego se pide el id de la estampilla, la fecha y el monto. Luego se obtiene el time_value original de la estampilla, que se le agrega o modifica el valor desiado, después se realiza la query que agrega el nuevo time_value.

Agregar:

```
Ingrese su opción: 6
(1) Agregar time_value
(2) Modificar time_value
Ingrese una opcion: 1
Ingrese id: 51a9c348-ea04-4e2c-822a-e879e22b851f
Ingrese fecha: 2024-11-11
Ingrese monto: 100
Nuevo registro agregado a time_value para la estampilla.
```

```
Estampilla encontrada - ID: 51a9c348-ea04-4e2c-822a-e879e22b851f
Datos: {
  condition: 'dañado',
  country: 'Chile',
  design: 'Imagen conmemorativa',
  face_value: 10,
  seller: 'Ana López',
  series: 'Bicentenario',
  status: 'disponible',
  tags: [ 'bicentenario', 'historia' ],
  time_value: [
    { date: '2024-11-09', value: 10 },
    { date: '2024-11-11', value: 100 }
  ],
  title: 'Estampilla Bicentenario',
  transaction_history: [ { date: '2024-11-09', action: 'venta' } ],
  year: 2010
```

Modificar:



```
Ingrese su opción: 6
(1) Agregar time_value
(2) Modificar time_value
Ingrese una opcion: 2
Ingrese id: de29222d-709b-43d0-a8e0-e9fcee0106a3
Ingrese fecha: 2024-11-11
Ingrese monto: 2
```

```
Datos: {
  condition: 'dañado',
  country: 'Chile',
  design: 'Imagen conmemorativa',
  face_value: 10,
  seller: 'Ana López',
  series: 'Bicentenario',
  status: 'disponible',
  tags: [ 'bicentenario', 'historia' ],
  time_value: [
    { date: '2024-11-09', value: 10 },
    { date: '2024-11-11', value: 2 }
  ],
  title: 'Estampilla Bicentenario',
  transaction_history: [ { date: '2024-11-09', action: 'venta' } ],
  year: 2010
}
```

8) (Item5) Comprar estampilla disponible:

```
async function comprar() {
  const title = await preguntar("Ingrese title: ");
  const seller = await preguntar("Ingrese seller: ");
  const querySelect = `
    SELECT stamp_id, status, transaction_history
    FROM stamps
    WHERE title = ? AND seller = ? ALLOW FILTERING;
  `;

  try {
    const result = await client.execute(querySelect, [title, seller], { prepare: true });
    if (result.rowLength === 0) {
      console.log('No se encontró la estampilla con el título y vendedor especificados.');
      return;
    }
    const stamp = result.rows[0];
    if (stamp.status !== 'disponible') {
      console.log('La estampilla no está disponible para la compra.');
      return;
    }
    if (!stamp.transaction_history) {
      stamp.transaction_history = [];
    }
    // Obtener la fecha actual
    const fechaCompra = new Date().toISOString().split('T')[0];
    }
    // Convertir la nueva transacción en una tupla
    const { Tuple } = require('cassandra-driver').types;
    const nuevaTransaccion = new Tuple(fechaCompra, 'compra');

    // Agregar la nueva transacción como una tupla
    stamp.transaction_history.push(nuevaTransaccion);
    // Realizar la transacción de compra
    const batchQuery = `
      BEGIN BATCH
        UPDATE stamps SET status = 'vendido' WHERE stamp_id = ?;
        UPDATE stamps SET seller = 'D' WHERE stamp_id = ?;
        UPDATE stamps SET transaction_history = ? WHERE stamp_id = ?;
      APPLY BATCH;
    `;
    await client.execute(batchQuery, [stamp.stamp_id, stamp.stamp_id, stamp.transaction_history, stamp.stamp_id], { prepare: true });
    console.log('Estampilla comprada exitosamente.');
  } catch (error) {
    console.error('Error al intentar comprar la estampilla:', error);
  }
}
```

La función realiza una “compra” de una estampilla especifica, primero se le pregunta al usuario el título de la estampilla y el vendedor, se verifica que está disponible y con una batchquery se realizan los cambios para efectuar la compraventa de la estampilla, luego se muestra por pantalla un mensaje de confirmación.

```
Ingrese su opción: 7
Ingrese title: Estampilla de la Unión Europea
Ingrese seller: Luc Dupont
Estampilla comprada exitosamente.
```

```
Ingrese su opción: 3
Ingrese id: da5b76cf-fd9b-44ca-bebd-81e42e5aff52
Historial de transacciones: [
  Tuple { elements: [ '2024-05-10', 'venta' ], length: 2 },
  Tuple { elements: [ '2024-11-11', 'compra' ], length: 2 }
]
```

9) (Item6) Ocupando vista materializada, mostrar las estampillas más caras:

```
async function view() {
  //<--- Crear vista materializada --->
  const queryView = `
    CREATE MATERIALIZED VIEW IF NOT EXISTS most_expensive_stamp_by_condition AS
    SELECT stamp_id, title, country, year, face_value, condition, status, time_value
    FROM stamps
    WHERE condition IS NOT NULL AND stamp_id IS NOT NULL
    PRIMARY KEY (condition, stamp_id);
  `;
  try {
    await client.execute(queryView);
    console.log("Vista materializada creada correctamente.");
  } catch (error) {
    console.error("Error al crear la vista materializada:", error);
    return;
  }
  //<--- Buscar la más cara --->
  const year = await preguntar("Ingrese year: ");
  const queryNuevo = `
    SELECT stamp_id, title, face_value
    FROM most_expensive_stamp_by_condition
    WHERE condition = 'nuevo' AND year = ? ALLOW FILTERING;
  `;
```

```
try {
  const result = await client.execute(queryNuevo, [year], { prepare: true });
  if (result.rowLength > 0) {
    const mostExpensiveStamp = result.rows.reduce((max, row) => {
      return row.face_value > (max.face_value || 0) ? row : max;
    }, {});
    const stampId = mostExpensiveStamp.stamp_id ? mostExpensiveStamp.stamp_id.toString() : "ID no disponible";
    const { stamp_id, ...dataWithoutId } = mostExpensiveStamp;
    console.log(`La estampilla más cara en condición "nuevo" para el año ${year} es: ID: ${stampId}, Datos:`, dataWithoutId);
  } else {
    console.log(`No se encontraron estampillas en condición "nuevo" para el año ${year}.`);
  }
} catch (error) {
  console.error('Error al obtener la estampilla más cara:', error);
}
const queryUsado = `
  SELECT stamp_id, title, face_value
  FROM most_expensive_stamp_by_condition
  WHERE condition = 'usado' AND year = ? ALLOW FILTERING;
`;
```

```
try {
  const result = await client.execute(queryDanio, [year], { prepare: true });
  if (result.rowLength > 0) {
    const mostExpensiveStamp = result.rows.reduce((max, row) => {
      return row.face_value > (max.face_value || 0) ? row : max;
    }, {});
    const stampId = mostExpensiveStamp.stamp_id ? mostExpensiveStamp.stamp_id.toString() : "ID no disponible";
    const { stamp_id, ...dataWithoutId } = mostExpensiveStamp;
    console.log(`La estampilla más cara en condición "dañado" para el año ${year} es: ID: ${stampId}, Datos:`, dataWithoutId);
  } else {
    console.log(`No se encontraron estampillas en condición "dañado" para el año ${year}.`);
  }
} catch (error) {
  console.error('Error al obtener la estampilla más cara:', error);
}
}
```

La función crea una vista materializada y busca la estampilla más cara para cada condición en un año determinada. Primero se crea la vista, luego se le solicita al usuario un año para buscar, luego se busca para cada condición (nuevo, usado, dañado) la estampilla más cara utilizando una query. Luego se muestra en consola la información de cada estampilla.

```
Ingrese su opción: 8
Vista materializada creada correctamente.
Ingrese year: 2020
La estampilla más cara en condición "nuevo" para el año 2020 es: ID: 9e2a2151-456c-43b5-96f4-96a62f353ca8, Datos: { title: 'Estampilla de la Independencia', face_value: 5 }
La estampilla más cara en condición "usado" para el año 2020 es: ID: b8a6e15c-9e01-4200-933a-4aee1677460e, Datos: { title: 'Estampilla del Bicentenario de Argentina', face_value: 7.5 }
No se encontraron estampillas en condición "dañado" para el año 2020.
```

10) (Item7) Agregar una estampilla en la base de datos:

```
async function agregar() {
  const title = await preguntar("Ingrese title: ");
  const country = await preguntar("Ingrese country: ");
  const year = await preguntar("Ingrese year: ");
  const series = await preguntar("Ingrese series: ");
  const design = await preguntar("Ingrese desing: ");
  const face_value = await preguntar("Ingrese face_value: ");
  const condition = await preguntar("Ingrese condition: ");
  const status = await preguntar("Ingrese status: ");
  const seller = await preguntar("Ingrese seller: ");
  let flag = true;
  let tags = [];
  while(flag){
    console.log("(1) agregar un tag");
    console.log("(2) no agregar más tags");
    let quiere_tag = await preguntar("Ingrese su opcion: ");
    if(quiere_tag == "1"){
      let tag = await preguntar("Ingrese tag: ");
      tags.push(tag);
    }
    else if(quiere_tag == "2"){
      flag = false;
    }
  }
```

```
  const query = `
  INSERT INTO stamps (stamp_id, title, country, year, series, design, face_value, condition, status, seller, transaction_history, tags, time_value)
  VALUES (uuid(), ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
  `;

  try{
    await client.execute(query, [title, country, year, series, design, face_value, condition, status, seller, [], tags, []], { prepare: true });
    console.log('Estampilla añadida con éxito');
  } catch (error) {
    console.error('Error al añadir la estampilla:', error);
  }
}
```

La función permite agregar una estampilla a la base de datos, primero se le pregunta al usuario los datos de la estampilla, luego se realiza una query deg inserción con esos datos.

```
Ingrese su opción: 9
Ingrese title: Jordi Wild U de Chile
Ingrese country: Chile
Ingrese year: 2024
Ingrese series: famoso
Ingrese desing: famoso youtuber con polera de equipo de futbol
Ingrese face_value: 100
Ingrese condition: nuevo
Ingrese status: disponible
Ingrese seller: Crazy Boy
(1) agregar un tag
(2) no agregar más tags
Ingrese su opcion: 1
Ingrese tag: youtuber
(1) agregar un tag
(2) no agregar más tags
Ingrese su opcion: 2
```

```
Ingrese su opción: 1
Ingrese country: Chile
Ingrese year: 2024
Ingrese tag: youtuber
Estampilla encontrada - ID: d6d2af07-1868-4f9a-b9f7-9bd18e02f979, Datos: {
  condition: 'nuevo',
  country: 'Chile',
  design: 'famoso youtuber con polera de equipo de futbol',
  face_value: 100,
  seller: 'Crazy Boy',
  series: 'famoso',
  status: 'disponible',
  tags: [ 'youtuber' ],
  time_value: null,
  title: 'Jordi Wild U de Chile',
  transaction_history: null,
  year: 2024
```

11) (Item8) Ocupando SASI, mostrar las estampillas más baratas:

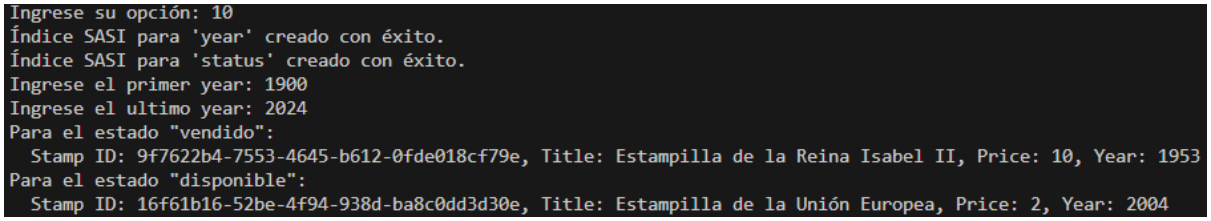
```
async function mas_barata() {
  //<--- Indice SASI --->
  const createYearIndexQuery = `
    CREATE CUSTOM INDEX IF NOT EXISTS ON stamps(year)
    USING 'org.apache.cassandra.index.sasi.SASIIndex'
    WITH OPTIONS = {
      'mode': 'PREFIX'
    };
  `;
  const createStatusIndexQuery = `
  CREATE CUSTOM INDEX IF NOT EXISTS ON stamps(status)
  USING 'org.apache.cassandra.index.sasi.SASIIndex';
  `;
  try {
    await client.execute(createYearIndexQuery);
    console.log("Índice SASI para 'year' creado con éxito.");

    await client.execute(createStatusIndexQuery);
    console.log("Índice SASI para 'status' creado con éxito.");
  } catch (error) {
    console.error("Error al crear los índices SASI:", error);
  }
```

```
const startYear = await preguntar("Ingrese el primer year: ")
const endYear = await preguntar("Ingrese el ultimo year: ")
const query = `
  SELECT stamp_id, title, face_value, status, year
  FROM stamps
  WHERE year >= ? AND year <= ?
  ALLOW FILTERING
`;
try {
  const result = await client.execute(query, [startYear, endYear], { prepare: true });
  // Agrupar las estampillas por 'status'
  const groupedByStatus = result.rows.reduce((acc, row) => {
    // Si el status no existe en el acumulador, lo agregamos
    if (!acc[row.status]) {
      acc[row.status] = [];
    }
    acc[row.status].push(row);
    return acc;
  }, {});
  // Iterar sobre cada grupo de 'status' y encontrar la estampilla más barata
  for (const status in groupedByStatus) {
    if (groupedByStatus[status].length > 0) {
      // Ordenar las estampillas de este estado por 'face_value' en orden ascendente
      const cheapestStamp = groupedByStatus[status]
        .sort((a, b) => a.face_value - b.face_value)[0];

      // Mostrar la estampilla más barata para este estado
      console.log(`Para el estado "${status}":`);
      console.log(`  Stamp ID: ${cheapestStamp.stamp_id.toString()}, Title: ${cheapestStamp.title}, Price: ${cheapestStamp.face_value}, Year: ${cheapestStamp.year}`);
    } else {
      console.log(`No se encontraron estampillas para el estado "${status}".`);
    }
  }
} catch (error) {
  console.error('Error al ejecutar la consulta:', error);
}
```

La función crea el índice SASI, para luego mostrar la estampilla más barata de cada condición en un rango de años. Primero se le pide al usuario los años en que se buscarán las estampillas, luego se realiza la query que busca la estampilla más barata, luego se muestra por consola el resultado.



12) (Item9) Técnicas para garantizar la consistencia entre tablas:

- Usar vista materializada: crea proyecciones de datos de una tabla en otra vista, las cuales se mantienen automáticamente sincronizadas a la tabla base con la nueva vista. Esto ayuda a mantener las tablas actualizadas sin tener que hacerlo manualmente.
- Batch statements: esta técnica permite agrupar operaciones de inserción, eliminación y actualización para más de una tabla. El sistema agrupa estos comandos en un solo bloque, luego intenta ejecutarlos como una unidad. Si una de las operaciones falla, intentará revertir todo el bloque (batch). Mantiene consistencia para varios cambios de datos, especial cuando estos dependen uno de otros.