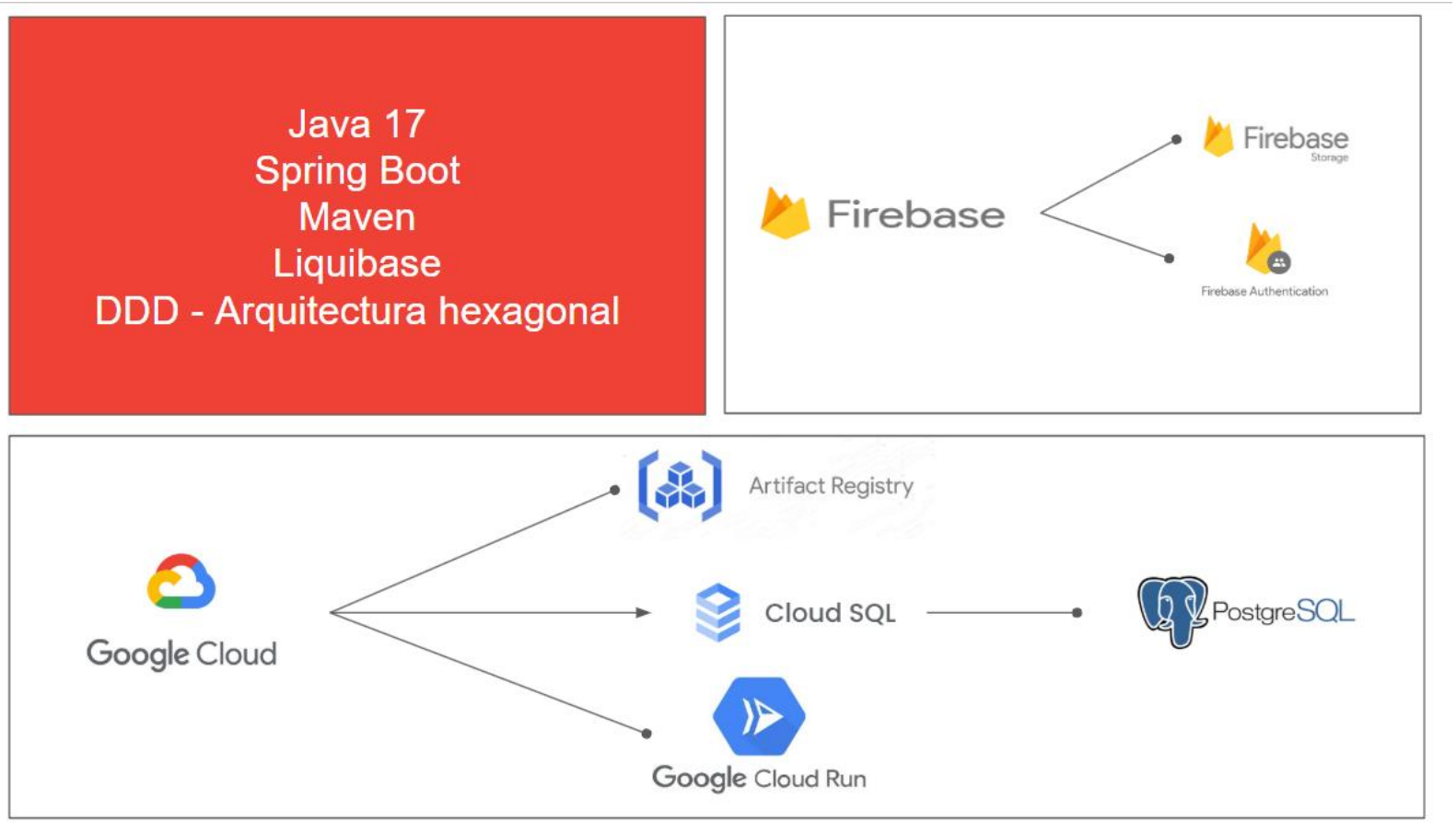
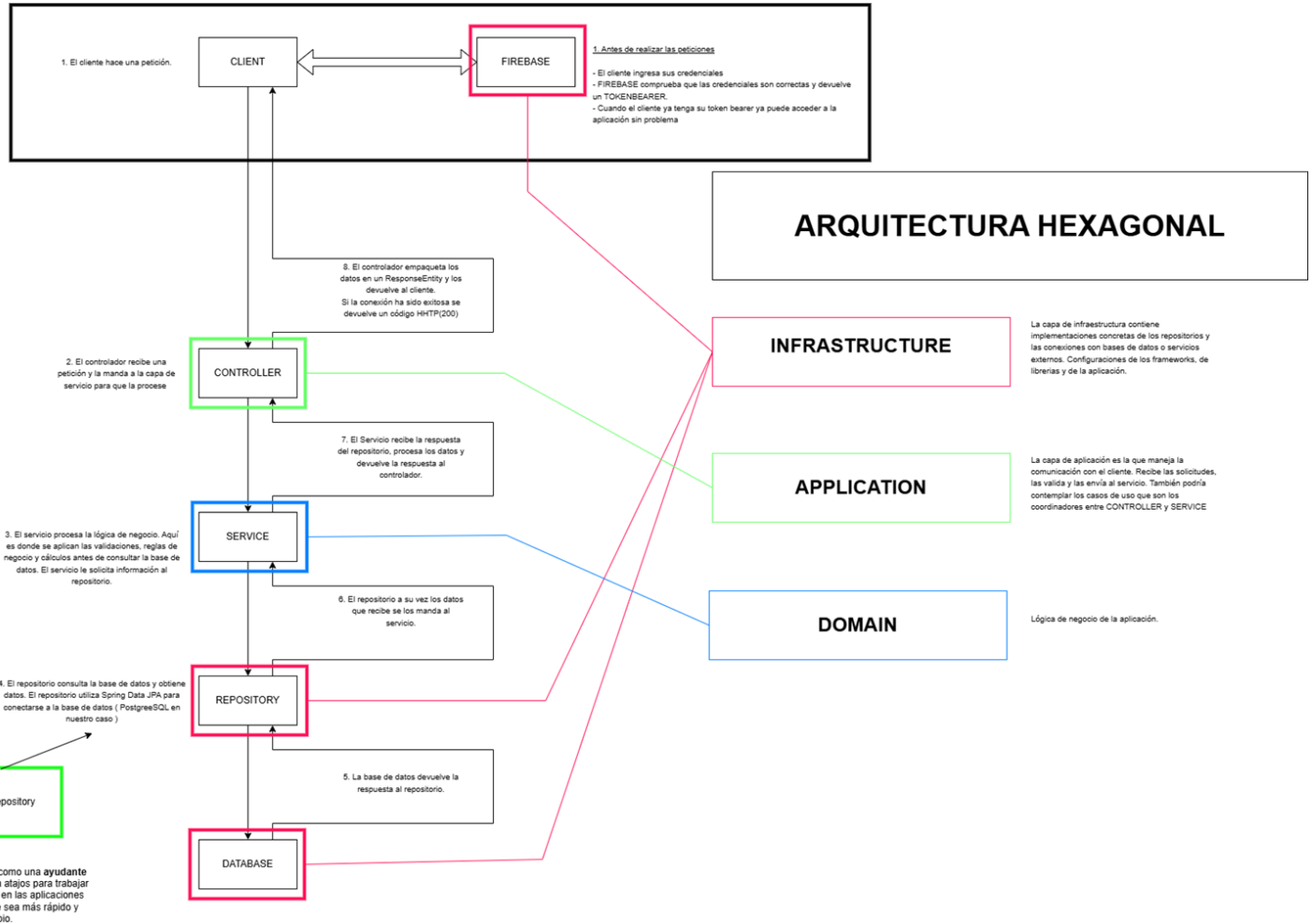
Politikea

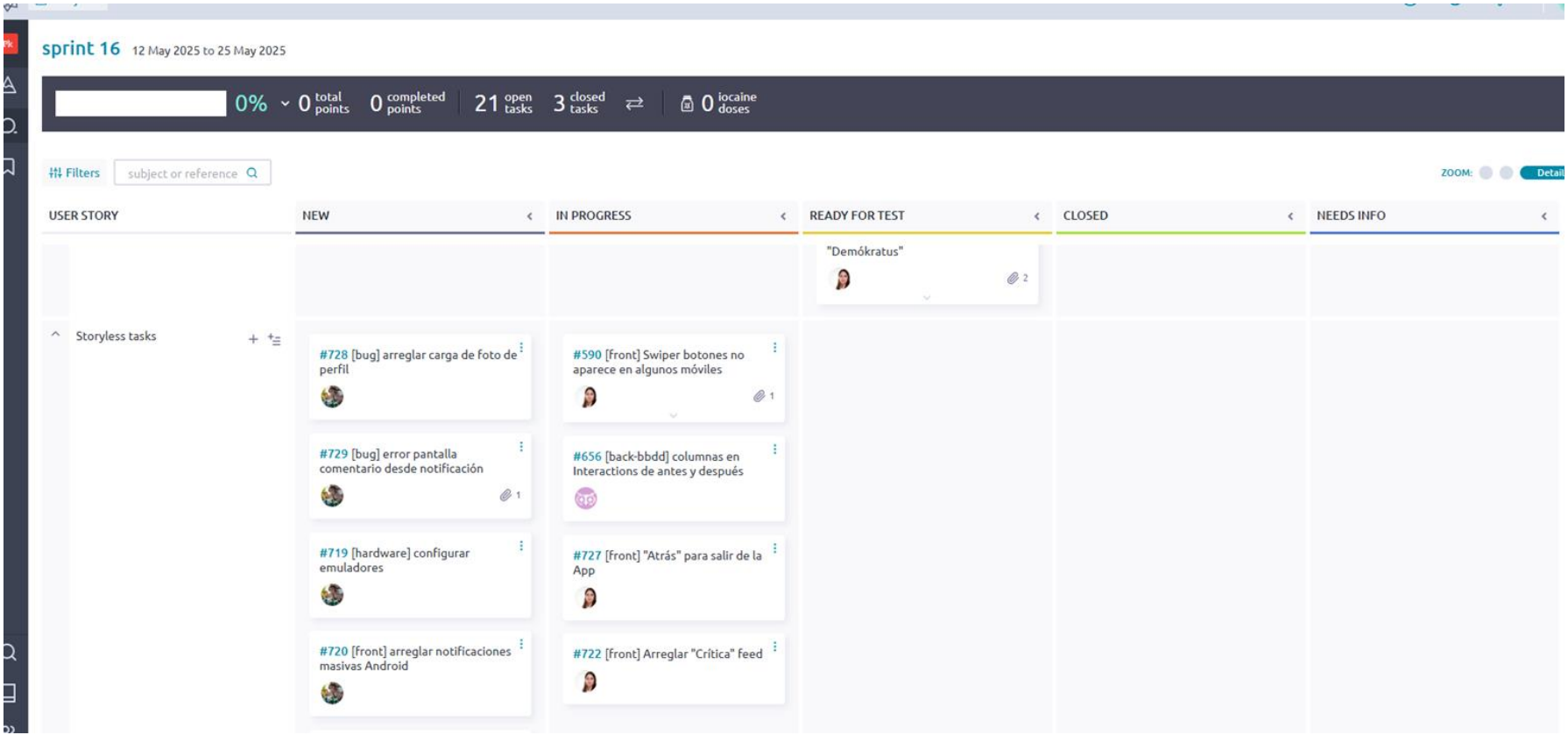
Impulsando una democracia sana y funcional mediante una participación política accesible, informada y reflexiva

Funcionamiento de la aplicación.



Flujo de la aplicación.



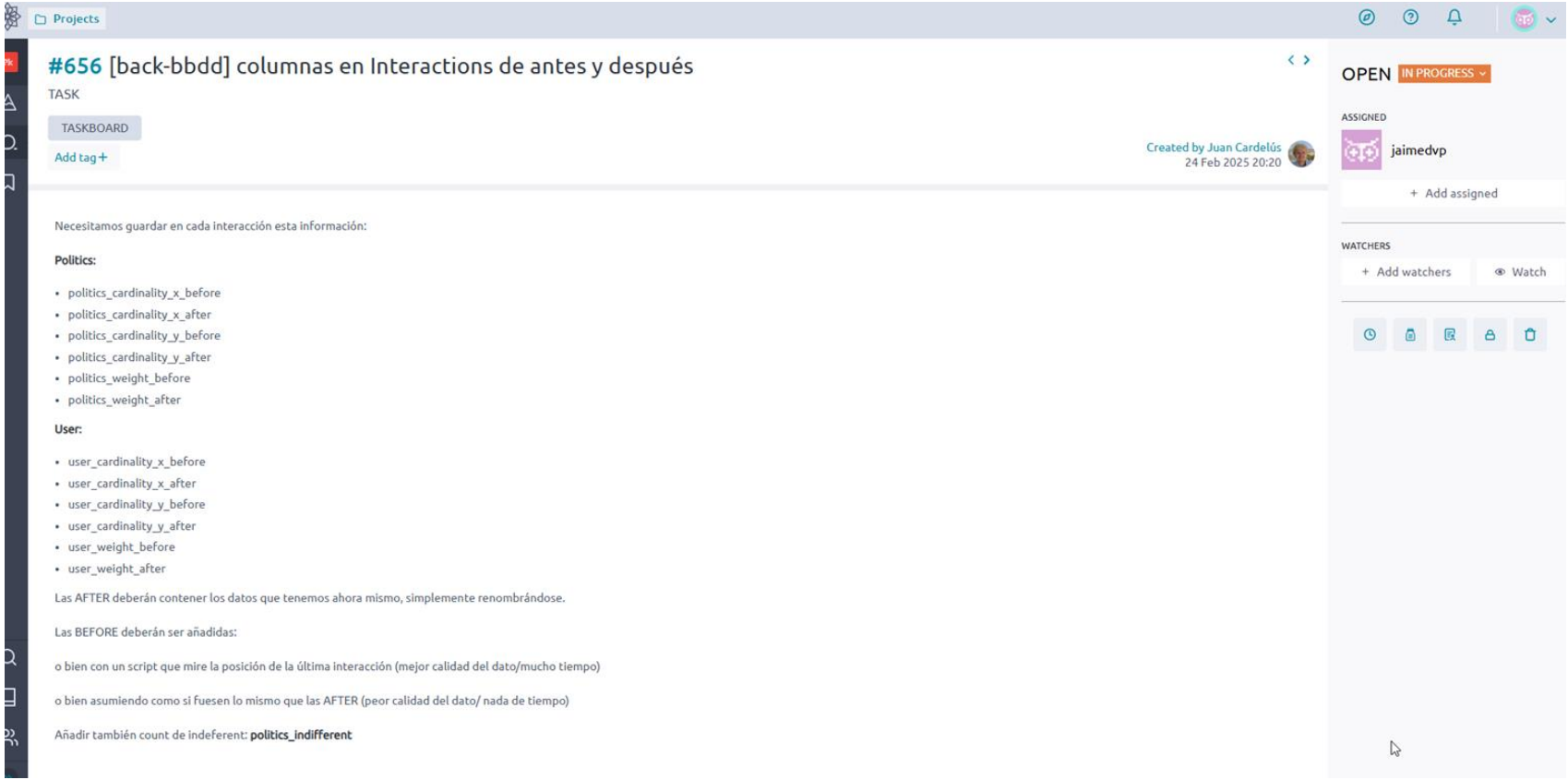

sprint 16 12 May 2025 to 25 May 2025
0% 0 total points 0 completed points 21 open tasks 3 closed tasks 0 iocaine doses
Filters
subject or reference
ZOOM:
Detail
USER STORY
NEW
IN PROGRESS
READY FOR TEST
CLOSED
NEEDS INFO
"Demókratus"
Storyless tasks
#728 [bug] arreglar carga de foto de perfil
#729 [bug] error pantalla comentario desde notificación
#719 [hardware] configurar emuladores
#720 [front] arreglar notificaciones masivas Android
#590 [front] Swiper botones no aparece en algunos móviles
#656 [back-bbdd] columnas en Interactions de antes y después
#727 [front] "Atrás" para salir de la App
#722 [front] Arreglar "Crítica" feed

1. Crear tu rama, nomenclatura de gitflow
   -> feature/ nuevos evolutivos
   -> bug/ corrección de errores
   -> fix/ "mini" corrección de errores
   -> hotfix/ -> "mini" corrceción de un error directamente a producción

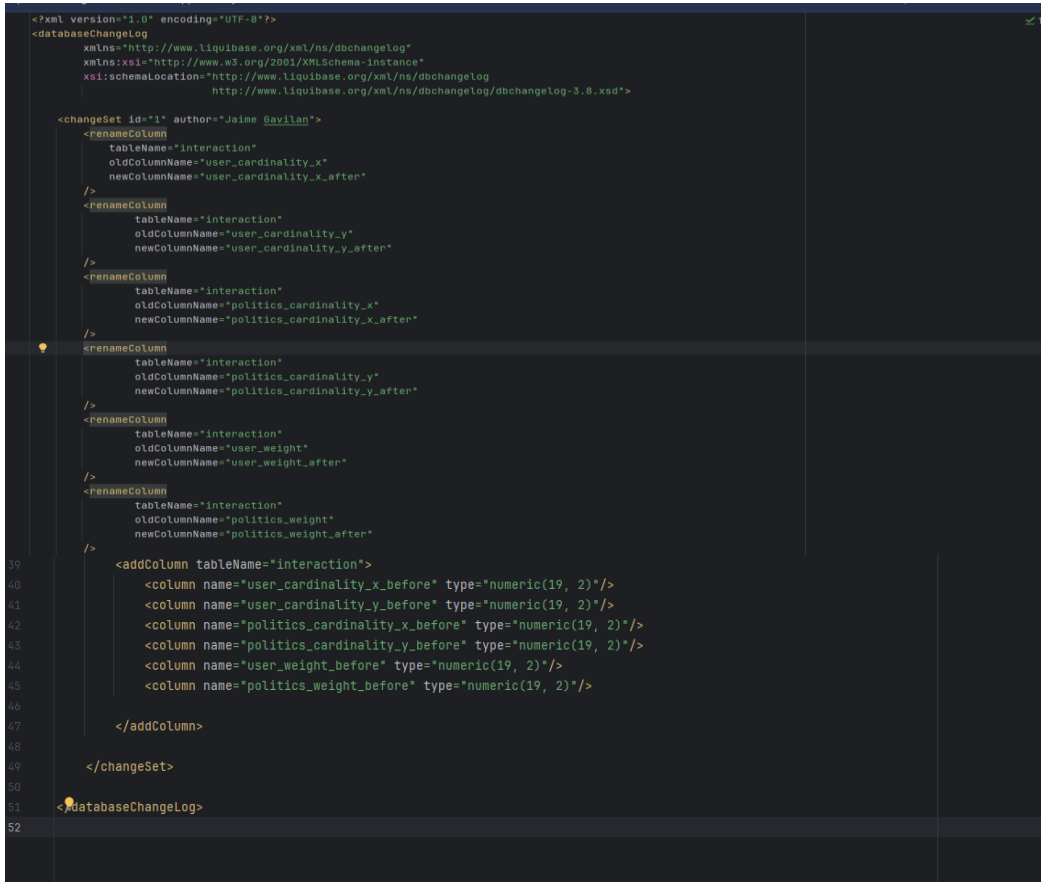
   ejemplo: feature/HUXX-T656
2. hacer tarea (poner in-progress en taiga)

   En este caso:
      1. Cambio en la base de datos creando un nuevo changelog y añadiendolo al changelog master
      2. Modificar el DBO de la tabla (en este caso InteractionDbo
      3. Modifica Model
      4. Modificar métodos/funciones de los servicios de user y politics en los métodos "interact" que deben guardar los nuevos datos.
3. subir rama
4. Crear merge request

Tareas realizadas

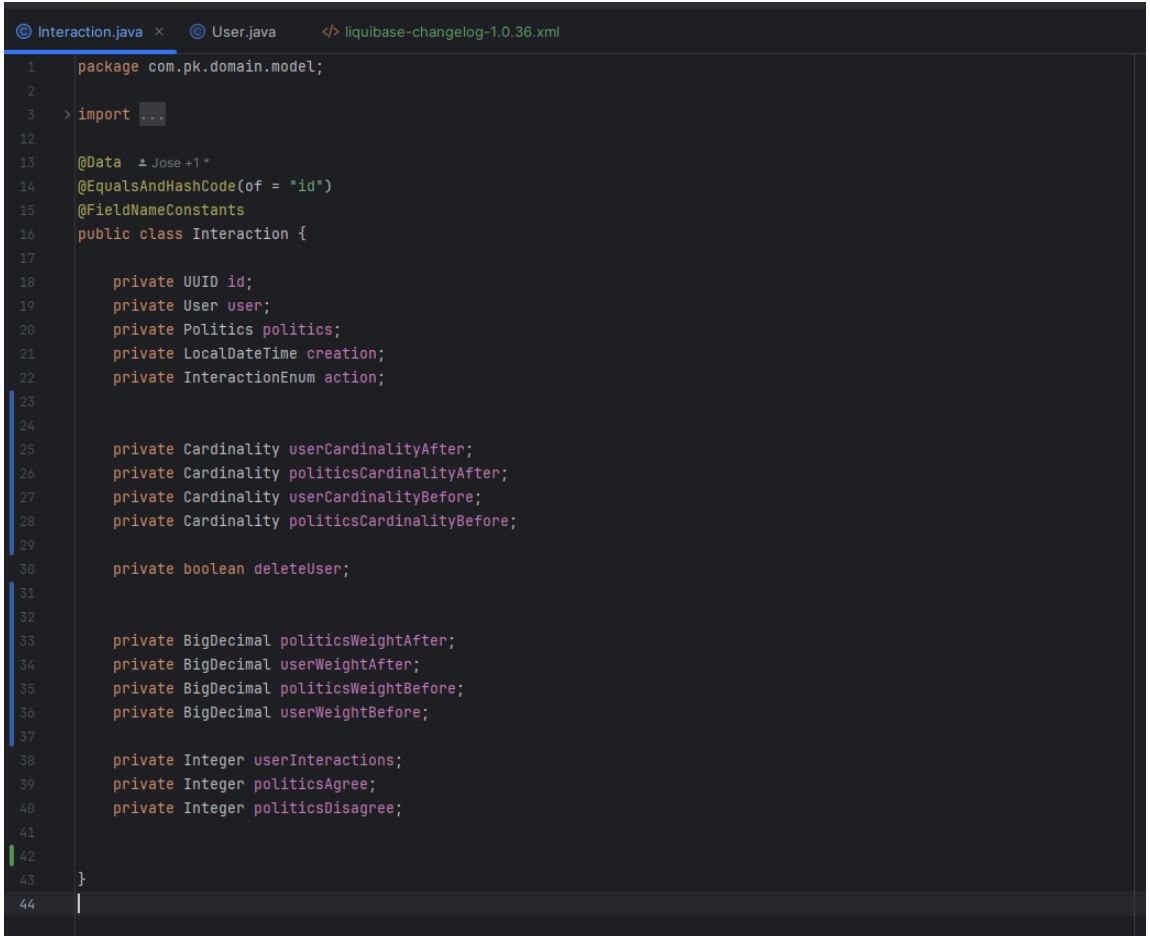


Crear un nuevo databaseChangeLog

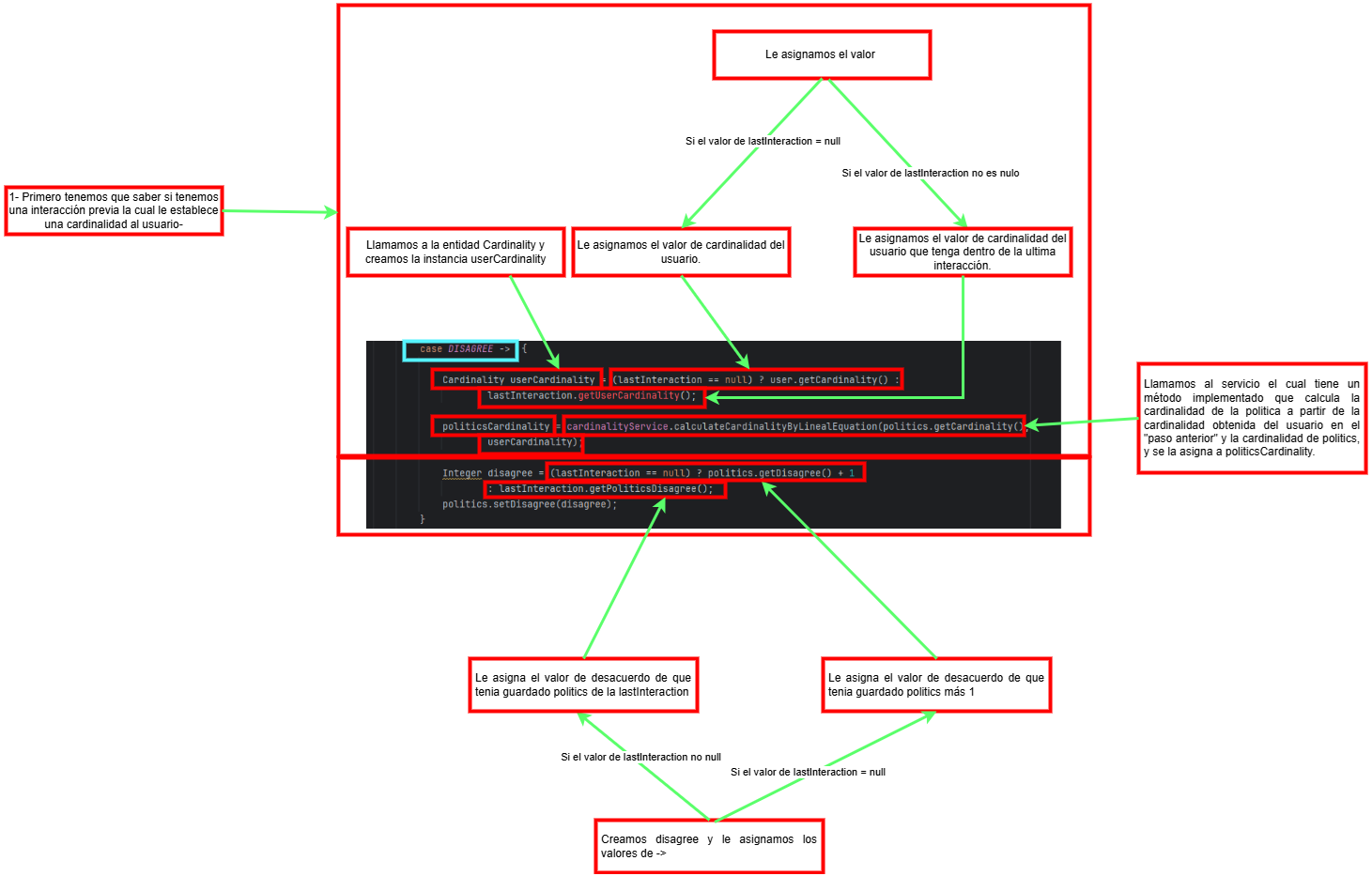


```xml
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
        xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
                            http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.8.xsd">

    <changeSet id="1" author="Jaime Gavilan">
        <renameColumn
            tableName="interaction"
            oldColumnName="user_cardinality_x"
            newColumnName="user_cardinality_x_after"
        />
        <renameColumn
                tableName="interaction"
                oldColumnName="user_cardinality_y"
                newColumnName="user_cardinality_y_after"
        />
        <renameColumn
                tableName="interaction"
                oldColumnName="politics_cardinality_x"
                newColumnName="politics_cardinality_x_after"
        />
        <renameColumn
                tableName="interaction"
                oldColumnName="politics_cardinality_y"
                newColumnName="politics_cardinality_y_after"
        />
        <renameColumn
                tableName="interaction"
                oldColumnName="user_weight"
                newColumnName="user_weight_after"
        />
        <renameColumn
                tableName="interaction"
                oldColumnName="politics_weight"
                newColumnName="politics_weight_after"
        />
        <addColumn tableName="interaction">
            <column name="user_cardinality_x_before" type="numeric(19, 2)"/>
            <column name="user_cardinality_y_before" type="numeric(19, 2)"/>
            <column name="politics_cardinality_x_before" type="numeric(19, 2)"/>
            <column name="politics_cardinality_y_before" type="numeric(19, 2)"/>
            <column name="user_weight_before" type="numeric(19, 2)"/>
            <column name="politics_weight_before" type="numeric(19, 2)"/>

        </addColumn>

    </changeSet>

</databaseChangeLog>
```

Modificar el model de la tabla Interaction



Ejemplo de la lógica de negocio de las políticas, en las interacciones que se producen.



Ejemplo de la lógica de negocio de las políticas, en las interacciones que se producen.

NÚMERO MÍNIMO PARA CALCULAR EL PESO	Aquí el código obtiene un valor mínimo llamado `minNumberAgreeForWeight` desde las configuraciones del sistema. Este valor representa la cantidad mínima de interacciones de "acuerdo" necesarias para calcular el peso. *"Para que este cálculo funcione, necesito al menos X acuerdos".*
CÁLCULO DEL PESO AFECTADO	El código verifica si los acuerdos (`politics.getAgree()`) superan el número mínimo requerido (`minNumberAgreeForWeight`). - **Si hay suficientes acuerdos:** Calcula un peso afectado (`affectedWeight`) usando el método `calculateWeightCircumference`). - **Si no hay suficientes acuerdos:** Establece el peso afectado como `BigDecimal.ONE`, que equivale a 1. "Si hay suficientes acuerdos, el peso afectado se calcula; si no, lo dejamos en 1 por defecto."
CARDINALIDAD DEL USUARIO	Es como decidir si tomamos el estado del usuario actual o el estado basado en su última interacción.
CÁLCULO DEL PESO POLÍTICO	Decide si se calcula un nuevo peso político o se usa uno ya guardado
PESO MÍNIMO AFECTANTE	El código obtiene un valor fijo llamado **peso mínimo afectante** (`MINIMUM_AFFECTING_WEIGHT`) desde las configuraciones del sistema. Este valor garantiza que el peso político nunca sea menor que este límite.
INTERACCIONES TOTALES Y CONCENSO	Calcula la cantidad total de interacciones, sumando los acuerdos (`politics.getAgree()`) y desacuerdos (`politics.getDisagree()`). Obtiene el número mínimo de interacciones necesarias para calcular la variación del consenso desde las configuraciones (`CONSENSUS_VARIATION_MINIMUM`). "Para calcular el consenso, necesito al menos X interacciones en total."
CÁLCULO DE VARIACIÓN DEL CONCENSO	- Obtiene la **pendiente de variación del consenso** (`CONSENSUS_VARIATION_SLOPE`). - Calcula el porcentaje de consenso como la proporción de acuerdos sobre el total de interacciones. - Ajusta el peso político (`affectingWeight`) usando un coeficiente de variación que depende del consenso.
AJUSTE DEL PESO POLÍTICO MÍNIMO	El código asegura que el peso político nunca sea menor que el peso mínimo afectante (`minimunAffectingWeight`).
LÍMITE MÍNIMO DE INTERACCIONES	- **Límite de interacciones permitidas** (`CALCULATE_CARDINALITY_POLITICS_LIMIT_INTERACTIONS`). - **Cantidad mínima de interacciones necesarias** (`CALCULATE_CARDINALITY_POLITICS_MIN_INTERACTIONS`). Son límites que ayudan a que el cálculo sea más controlado
AJUSTE DE LAS INTERACCIONES	- Usa los **acuerdos** actuales o de la última interacción. - Limita las interacciones al máximo permitido, si exceden. - Las eleva al mínimo requerido, si son menores.
CÁLCULO DE LA CARDINALIDAD POLÍTICA	Usa la cardinalidad política, la cardinalidad del usuario, los pesos calculados y las interacciones para calcular la nueva **cardinalidad política**. Incrementa el número de acuerdos en 1 (`agree = agree+1`) y actualiza el registro de acuerdos en las políticas. Es el paso final donde todo lo calculado se aplica y se registra para futuras interacciones.

```
case AGREE -> {
    Integer minNumberAgreeForWeight =
            politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.MIN_PUBLIC_POLITICS_INTERACTIONS_AGREE_FOR_WEIGHT)
                    .getValueInteger();

    BigDecimal affectedWeight = (minNumberAgreeForWeight < politics.getAgree()) ?
            spectrumService.calculateWeightCircumference(politics.getCardinality()) :
                BigDecimal.ONE;

    Cardinality userCardinality = (lastInteraction == null) ? user.getCardinality() :
            lastInteraction.getUserCardinality();

    BigDecimal affectingWeight = (lastInteraction == null) ?
            spectrumService.calculateWeightCircumference(user.getCardinality()) :
            lastInteraction.getUserWeight();

    BigDecimal minimunAffectingWeight = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.MINIMUM_AFFECTING_WEIGHT)
            .getValueBigDecimal();

    int totalInteractions = politics.getAgree() + politics.getDisagree();

    Integer minimumInteractions = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CONSENSUS_VARIATION_MINIMUM)
            .getValueInteger();

    if (totalInteractions >= minimumInteractions) {

        Double consensusVariationSlope = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CONSENSUS_VARIATION_SLOPE)
                .getValueDouble();

        double consensusPercentage = (double) politics.getAgree() / totalInteractions;

        double consensusCoefficientVariation = Math.log(Math.pow(1.00-consensusPercentage, consensusVariationSlope)+1) / Math.log(Math.pow(0.50, consensusVariationSlope)+1);
        affectingWeight = affectingWeight.multiply(BigDecimal.valueOf(consensusCoefficientVariation));
    }

    affectingWeight = affectingWeight.max(minimunAffectingWeight);

    Integer limitNumberInteraction =
            politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CALCULATE_CARDINALITY_POLITICS_LIMIT_INTERACTIONS)
                    .getValueInteger();

    Integer minNumberInteraction =
            politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CALCULATE_CARDINALITY_POLITICS_MIN_INTERACTIONS)
                    .getValueInteger();

    Integer agree = (lastInteraction == null) ? politics.getAgree()
            : lastInteraction.getPoliticsAgree();

    Integer interactions = agree;

    if (interactions > limitNumberInteraction) {
        interactions = limitNumberInteraction;
    }

    if (agree < minNumberInteraction) {
        interactions = minNumberInteraction;
    }

    politicsCardinality = cardinalityService.calculateCardinalityConservativeMotion(politics.getCardinality(),
            userCardinality, affectedWeight, affectingWeight, interactions);

    agree = agree+1;
    politics.setAgree(agree);
```

Ejemplo de la lógica de negocio de los usuarios, en las interacciones que se producen.

DEFINICION DE POLITICS CARDINALITY	Si lastInteraction es NULL cogemos la cardinalidad del objeto política. si lastinteraction no es NULL cogemos la cardinalidad de lastInteraction.
CÁLCULO DE USER CARDINALITY	Llama a un método (`calculateCardinalityByLinealEquation`) del servicio `cardinalityService`. Este método toma dos parámetros: - La cardinalidad del usuario (`user.getCardinality()`). - La cardinalidad política calculada previamente (`politicsCardinality`) Ajustar o recalcular la cardinalidad del usuario basado en una ecuación lineal.
CÁLCULO DE USER CARDINALITY	Aquí se obtienen las interacciones del usuario
CÁLCULO DE USER CARDINALITY	Usa el método `calculateWeightCircumference` del servicio `spectrumService`. Este método calcula un "peso" en la cardinalidad del usuario (`user.getCardinality()`).

```
case DISAGREE ->  {
    Cardinality politicsCardinality = (lastInteraction == null) ? politics.getCardinality() :
            lastInteraction.getPoliticsCardinality();
    userCardinality = cardinalityService.calculateCardinalityByLinealEquation(user,getCardinality(),
            politicsCardinality);
    userInteractions = user.getInteractions();
    userWeight = spectrumService.calculateWeightCircumference(user.getCardinality());
```

Se define una cardinalidad política según la última interacción.
Se calcula una nueva cardinalidad para el usuario basándose en la ecuación proporcionada.
Se obtienen las interacciones del usuario.
Se calcula un "peso" basado en la cardinalidad del usuario.

Ejemplo de la lógica de negocio de los usuarios, en las interacciones que se producen.

CÁLCULO DEL PESO AFECTADO	Tomamos la **cardinalidad del usuario** y calculamos un peso afectado utilizando el método `calculateWeightCircumference`. Este resultado se guarda como `userWeight`. La cardinalidad del usuario es su posición dentro del grupo en el que está posicionado y el peso afectado es impacto que tiene esa posición en los datos actuales del usuario.
CARDINALIDAD POLÍTICA	Si no hay una interacción registrada, se toma la cardinalidad de las políticas generales. Si hay una interacción registrada se usa la que está asignada a esa interacción.
PESO POLÍTICO	En esta parte, se define un valor llamado **peso político**. Si no hay interacción previa (`lastInteraction == null`), se calcula el peso político basándose en las reglas generales (`politics.getCardinality`). Si hay interacción previa, toma el peso político registrado en esa interacción.
VALOR MÍNIMO Y DIVISOR	**Peso mínimo afectante** (`MINIMUM_AFFECTING_WEIGHT`): el valor mínimo que puede tener el peso político. **Divisor de peso** (`POLITICS_WEIGHT_DIVISOR`): un número que sirve para ajustar el peso político en ciertos casos.
VALIDACIÓN DEL CUADRANTE	Se verifica si la **cardinalidad del usuario** está en el mismo cuadrante que la cardinalidad política. Si no están en el mismo cuadrante, ajusta el peso político dividiéndolo por el divisor y asegurándose de que no sea menor al peso mínimo "Si están en áreas diferentes, reduce el impacto político para evitar errores grandes."
INTERACCIONES MÍNIMAS	El código calcula cuántas interacciones totales han tenido los usuarios en términos de acuerdo (`Agree`) y desacuerdo (`Disagree`). Luego, verifica un valor mínimo necesario para que se considere un cálculo de consenso. "Si no hay suficientes interacciones, no podemos calcular consensos correctamente."
VALIDACIÓN DEL CONCENSO	Aquí se calcula cómo afecta el **consenso** (si la mayoría está de acuerdo). Si más del 50% de las interacciones están de acuerdo, ajusta el peso político basándose en un coeficiente matemático. "Es como ajustar el impacto según qué tan fuerte es el consenso"
	El peso afectante (`affectingWeight`) se asegura de no ser menor que el valor mínimo (`minimunAffectingWeight`)
LÍMITE DE INTERACCIONES	El código limita el número de interacciones de un usuario. Si superan un límite establecido en la configuración, las reduce al máximo permitido. Es como poner un techo para evitar que ciertos cálculos se vuelvan desproporcionados.
ACTUALIZACIÓN DE LA CARDINALIDAD	Finalmente, el sistema calcula la nueva **cardinalidad del usuario** usando todos los valores ajustados (peso afectado, peso político, interacciones). Luego actualiza las interacciones del usuario y registra el peso político para esta interacción.

```
case AGREE -> {
    BigDecimal affectedWeight = spectrumService.calculateWeightCircumference(user.getCardinality());
    userWeight = affectedWeight;

    Cardinality politicsCardinality = (lastInteraction == null) ? politics.getCardinality() :
            lastInteraction.getPoliticsCardinality();

    BigDecimal affectingWeight = (lastInteraction == null) ?
            spectrumService.calculateWeightCircumference(politics.getCardinality()) :
            lastInteraction.getPoliticsWeight();

    BigDecimal minimunAffectingWeight = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.MINIMUM_AFFECTING_WEIGHT)
            .getValueBigDecimal();

    BigDecimal divisor = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.POLITICS_WEIGHT_DIVISOR)
            .getValueBigDecimal();

    if (Boolean.FALSE.equals(cardinalityService.isCardinalityInQuadrant(user.getCardinality(), politicsCardinality))) {
        affectingWeight = affectingWeight.divide(divisor, 2, RoundingMode.HALF_UP);
        affectingWeight = affectingWeight.max(minimunAffectingWeight);
    }

    int totalInteractions = politics.getAgree() + politics.getDisagree();

    Integer minimumInteractions = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CONSENSUS_VARIATION_MINIMUM)
            .getValueInteger();

    if (totalInteractions >= minimumInteractions) {

        Double consensusVariationSlope = politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CONSENSUS_VARIATION_SLOPE)
                .getValueDouble();

        double consensusPercentage = (double) politics.getAgree() / totalInteractions;

        if (consensusPercentage > 0.5) {
            double consensusCoefficientVariation = Math.log(Math.pow(1.00 - consensusPercentage, consensusVariationSlope) + 1) / Math.log(Math.pow(0.50, consensusVariationSlope) + 1);
            affectingWeight = affectingWeight.multiply(BigDecimal.valueOf(consensusCoefficientVariation));
        }

    }

    affectingWeight = affectingWeight.max(minimunAffectingWeight);

    Integer limitNumberInteraction =
            politikeaSettingsService.getSettingsByName(PolitikeaSettingsEnum.CALCULATE_CARDINALITY_USER_LIMIT_INTERACTIONS)
                    .getValueInteger();

    //Usuario 100 interacciones.
    Integer numberInteractions = (lastInteraction == null) ?
            user.getInteractions() :
            lastInteraction.getUserInteractions();

    if (numberInteractions > limitNumberInteraction) {
        numberInteractions = limitNumberInteraction;
    }

    userCardinality = cardinalityService.calculateCardinalityConservativeMotion(user.getCardinality(),
            politicsCardinality, affectedWeight, affectingWeight, numberInteractions);

    userInteractions = user.getInteractions() + 1;

    interaction.setPoliticsWeight(affectingWeight);
```

Cloud Code

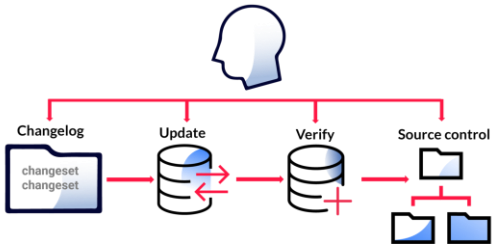
spring boot

Spring Data JPA

Liquibase

Project Lombok

Firebase



HIBERNATE

Maven

Swagger

Para finalizar la presentación es muy recomendable que me sigan en linkedIn y que por supuesto se descarguen **Politikea**!

LinkedIn

Politikea