# Descripción de la ampliación

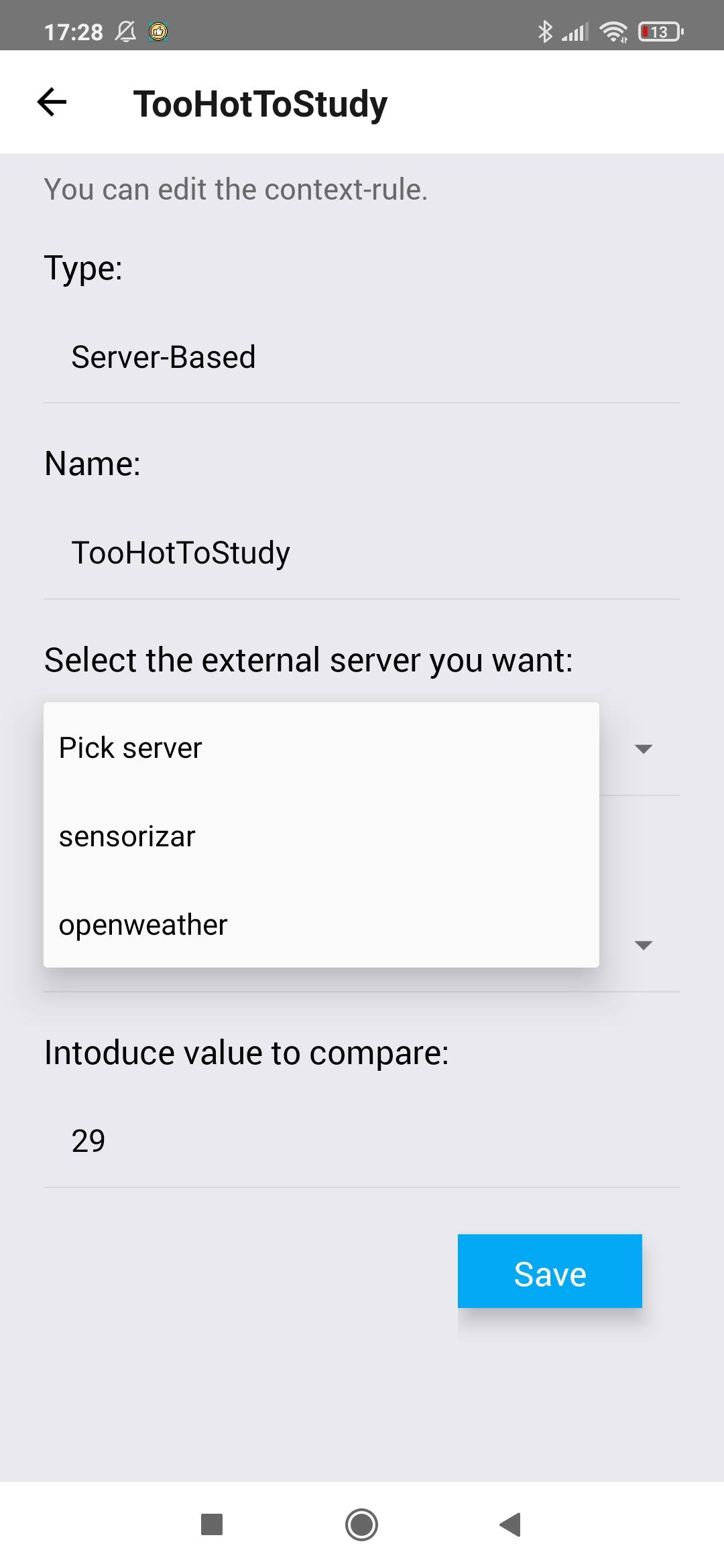
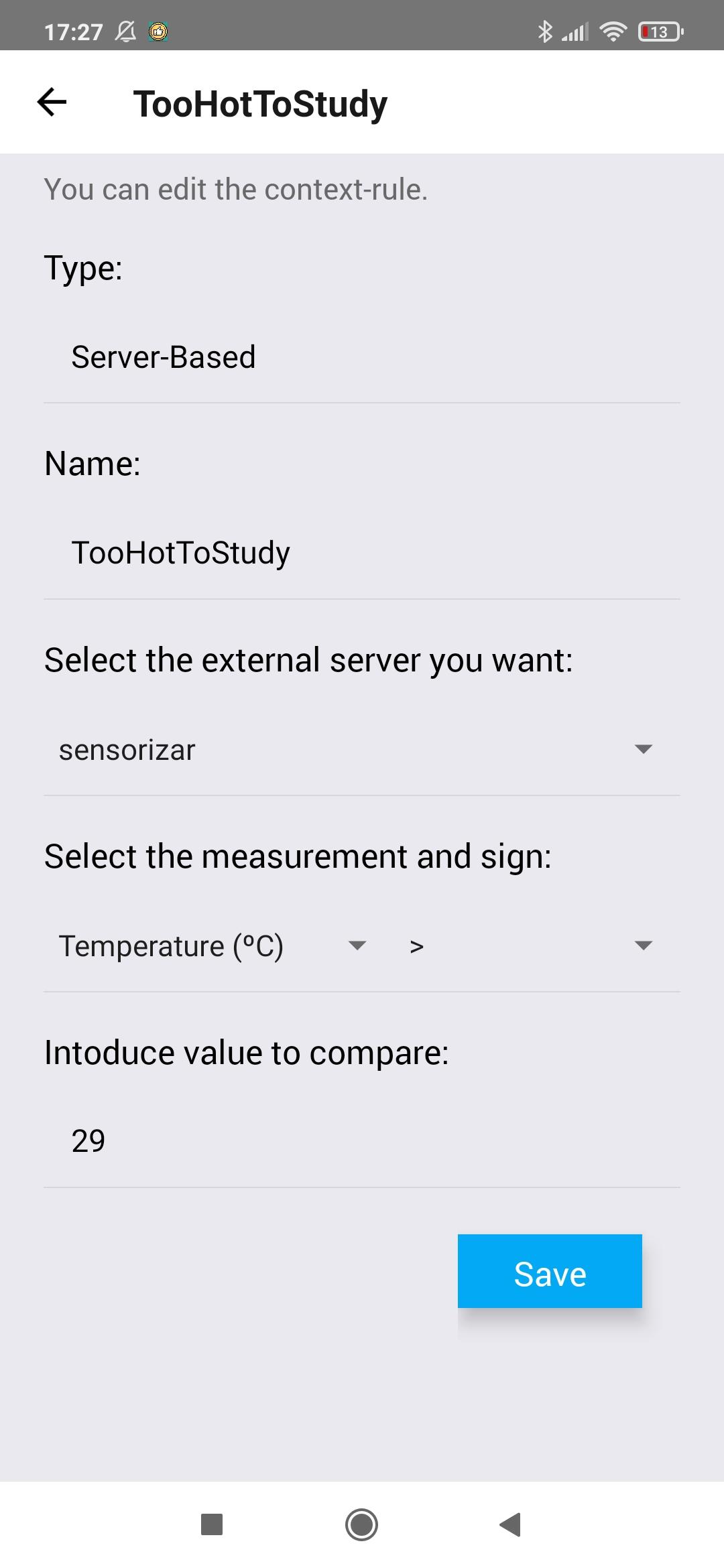
El sistema con el que se ha trabajado, en un principio contaba con **cuatro** tipos de *context-rules*: time-based, calendar-based, weather y location. Todas estas se podían combinar y también negar para formar otro tipo de regla, las *triggering rules* que activaban un tipo de recomendación (restaurantes, alojamientos…).

En la ampliación se ha desarrollado un nuevo tipo de *context-rule* mediante la cuál la aplicación se conecta con un servidor externo para consultar cierta información con la que se definirán las condiciones de la nueva regla. La incorporación de esta nueva regla nos permite definir otro escenario, donde el usuario estará en la universidad y el sistema le recomendará cuándo debe abandonar una habitación debido a parámetros como la humedad, co2 o temperatura, que serán consultadas a un servidor externo. Además recomendará otra nueva sala.

# Implementación de la nueva CR

El nuevo tipo de regla se llama **server-based context rule**. Al definirla, debemos indicar un nombre para la regla, el servidor externo al que queremos consultar la información, el parámetro que queremos tener en cuenta (humedad, temperatura, co2), el signo de la comparación y un valor concreto. En las **figuras 1 y 2** podemos ver un ejemplo de este tipo de regla.

Los servidores que aparecen como opciones al crear la regla están definidos en un fichero de configuración donde además se indican las peticiones que se van a utilizar de cada uno de ellos. En nuestro caso en el fichero de configuración hemos añadido 2 elementos: ***sensorizar*** (de la Universidad de Zaragoza que nos permite consultar información de las salas y clases según un identificador) y ***openweather***, que ya se utilizaba para implementar las *weather CR.*



Figuras 1 y 2: Ejemlo de *server-based context-rule*

En la figura 3 se puede observar un fragmento del fichero de configuración mencionado y la información que incluye sobre ***sensorizar***.



Figura 3: *external-requests.json*

Para obtener los datos de sensorizar se necesita realizar 2 peticiones, una para realizar un inicio de sesión y obtener un token, y otra para consultar la información utilizando ese token que hemos obtenido. El token tiene un tiempo de expiración de 2 horas y media, por lo tanto, almacenamos localmente (en realm, almacenamiento local ya mencionado en el tfg) el token con un timestamp que representa la hora a la que se ha obtenido el nuevo token. Antes de realizar la petición de consulta a sensorizar, se compara la hora actual con el timestamp y en caso de que la diferencia sea superior a 2.5 horas, se vuelve a realizar la petición de login para obtener un nuevo token.

# Implementación de las reglas Siddhi

Cuando definimos las reglas, se deben traducir a la sintaxis de Siddhi. La sintaxis en Siddhi de la regla mostrada en la figura 1 podemos verla:



Y negada:



La única diferencia entre la negada y la no negada es el signo. El ejemplo anterior tiene en cuenta la temperatura, pero con el co2 y la humedad sería el mismo formato:





Las negadas serían igual pero invirtiendo el signo.

Estas reglas se combinan con el resto de context rules para formar triggering rules de la misma manera que se explicó en el tfg, no presentan nada nuevo.

# Escenario de ejemplo

El escenario de prueba consiste en que Alice en esta ocasión tiene que estudiar pero no quiere estar en una clase con demasiada temperatura, co2 (ocupación) o humedad, porque puede dificultar la concentración para el estudio.

Alice define 3 context rules del tipo nuevo con las siguientes características:

- ***TooHotToStudy***: Server-Based con la temperatura

- ***TooHumidityToStudy***: Server-Based con la humedad

- ***BusyRoom:*** con CO2

Además, añade otras dos reglas de tipo *location* y *time-based*:

- ***Time4Lunch***: de tiempo con el intervalo de comer

- ***AtUniversity***: de localización, para indicar si está o no en la universidad

Actualmente, Alice tiene activas dos *triggering-rules:*

* ***ChangeRoomBcTooHot:*** TooHotToStudy, AtUniversity y time4lunch negada -> recomendar abandonar la sala y una nueva sala para acudir.
* ***ChangeRoomBcTooBusy:*** BusyRoom, AtUniversity y Time4Lunch negada -> recomendar abandonar la sala y una nueva sala para acudir.

Con estas reglas espera que nuestro sistema le recomiende una nueva sala cuando se cumplan las condiciones necesarias para cambiar de clase. La idea principal sería que desde Siddhi se detecte si hay que cambiar de sala y posteriormente, un recomendador nos indique que salas son las apropiadas con las condiciones requeridas. En nuestro caso, se recomienda otra sala de ejemplo. En el escenario, el usuario debe escanear un QR a la entrada de la clase para indicar al sistema que se encuentra en esa sala y pueda realizar las consultas necesarias a sensorizar teniendo en cuenta dónde está.

A partir de esto, se han realizado 5 contextos de prueba donde se pretende simular situaciones donde no se debe activar nada o donde debe activarse la recomendación de cambio de sala. Los contextos están el fichero *contextV2.json.*

Los resultados esperados son los siguientes:

* CONTEXTO 0: No hace nada
* CONTEXTO 1: Recomienda cambiar de sala porque la temperatura es **superior a 29**
* CONTEXTO 2: No hace nada porque está en el horario de comer (aunque la temperatura es superior a 29)
* CONTEXTO 3: Recomienda cambiar de sala porque la humedad es **superior a 750**
* CONTEXTO 4: No hace nada porque aunque la humedad y la temperatura son superiores, ya no se encuentra en la universidad

Los logs obtenidos siguen el mismo formato que el tfg y están en el fichero *logs.txt.*

Para los contextos 1 y 3 observamos que el resultado de Siddhi es *changeRoom*, justo lo esperado.

Las actividades de prueba creadas (salas) están en el fichero activties.json.





Para los contextos 0, 2 y 4 no se ha obtenido que hay que recomendar y por lo tanto el resultado ha sido correcto.