Expert Systems With Applications 242 (2024) 122714

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An approach for proactive mobile recommendations based on user-defined rules

Sergio Ilarri *, Raquel Trillo-Lado

Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna, 1, Zaragoza, 50018, Spain
I3A, Universidad de Zaragoza, Mariano Esquillor, s/n, Zaragoza, 50018, Spain

ARTICLE INFO

Dataset link: http://webdiis.unizar.es/~silarri/prot/RRules/

Keywords:
Mobile context-aware recommender systems
Context rules
Push-based recommendation
Data management
Complex event detection

ABSTRACT

In the Big Data era, context-aware mobile recommender systems are crucial in assisting citizens and tourists in making informed decisions, providing a suitable way for users to find the relevant data. These systems should be proactive, able to detect the ideal time and location to provide recommendations for a specific item or activity. To accomplish this, push-based recommender systems can be employed, utilizing context rules to determine when a recommendation should be initiated. However, there is very limited reported experience in defining and implementing such systems and a complete generic solution that adapts flexibly to the preferences of users and protects their privacy is still missing.

In this paper, we present a novel approach where appropriate types of recommendations are provided automatically, without the need for user input. Our proposal allows users to easily activate, deactivate, customize, and create rules for improved personalization. Additionally, the module that, based on the context, decides the types of recommendations required is executed on the user's mobile device, reducing wireless communication and safeguarding the user's privacy, as context data are evaluated locally. To illustrate the approach, we have developed R-Rules, a prototype for Android devices focused on the triggering of recommendation rules, which provides a friendly user interface that facilitates user personalization. We have evaluated various technological options and demonstrated the feasibility, performance, and scalability of the proposal, as well as its suitability to users' needs.

1. Introduction

Modern citizens face new challenges and effective data management is crucial in addressing these issues (Ilarri, 2021–2025; Ilarri et al., 2020). Software systems can play a vital role by supporting decision-making through the pre-selection of options relevant to a user, filtering out of the often-overwhelming set of choices. *Recommender Systems* (*RS*) are designed to learn user preferences and suggest items (such as points of interest, hotels, restaurants, etc.) or activities (such as visiting or consuming a specific item) that match those preferences (Lu et al., 2015; Park et al., 2012; Ricci et al., 2022).

The concept of *Context-Aware Recommender Systems –CARS–* (Adomavicius et al., 2011; Adomavicius & Tuzhilin, 2005, 2008; Colombo-Mendoza et al., 2017) expands the traditional $User \times Item$ approach of recommender systems to include a third dimension, which is the $Context$. This means that recommendations take into account the surroundings of the user (such as his/her location, the time, the weather, etc.), detected by the use of sensors of different type (Rault et al., 2017; Santos et al., 2010), to provide more relevant suggestions. When users are using a mobile device while on the move, proactive recommender systems (Hermoso et al., 2015) are typically favored over reactive recommender systems (del Carmen Rodríguez-Hernández & Ilarri, 2016); the former offer suggestions automatically, without the need for explicit user input, whereas the latter operate only upon user requests. This is particularly important in tourism scenarios (and mobile environments in general), where mobile users will be moving around with their mobile phones. For example, while visiting a city, it would be very interesting to have a mobile app that can suggest, in an unobtrusive way, appropriate items or activities to perform.

A fundamental component for creating a proactive recommender system is an intelligent decision module that determines when to trigger a recommendation based on the user's context. This helps to personalize the recommendations by adapting them to the individual's needs. For instance, if a tourist is exploring a city during lunchtime, the system might suggest a restaurant. To decide when a certain type of recommendations should be provided, the decision-making process can be based on context rules defined by the user. Although several

* Corresponding author at: Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna, 1, Zaragoza, 50018, Spain.
E-mail addresses: silarri@unizar.es (S. Ilarri), raqueltl@unizar.es (R. Trillo-Lado).

https://doi.org/10.1016/j.eswa.2023.122714
Received 8 June 2023; Received in revised form 9 November 2023; Accepted 23 November 2023
Available online 27 November 2023

technologies can be utilized, no recent studies have compared these options and proposed a full-fledged detailed solution for recommender systems. Besides, typically, these rule-based engines are run on servers rather than on the mobile devices of the users, which has important disadvantages, for example from the perspective of user privacy. As emphasized in Zhang and Sundar (2019), a reactive system constantly calling for user action can adversely affect the user experience, which can be addressed with a proactive approach, but at the same time the user privacy should be conveniently protected.

The main objective tackled in this paper is to define and implement a suitable approach to provide push-based recommendations in mobile environments, with a focus on a proactive architecture and the definition of rules that capture the user preferences concerning the behavior of the system. In particular, the main contributions are:

(a) *Architecture*. We have defined a generic architecture for mobile push-based recommendations, which includes a decision module executing on the mobile device of the user in order to efficiently decide when specific types of recommendations should be provided. This client-side recommendation triggering approach offers key advantages, from both the perspective of privacy preservation and wireless communication savings. To the best of our knowledge, there is no other generic architecture addressing these key issues for proactive mobile recommenders.
(b) *Flexible rules*. We have defined different types of rules (context rules and recommendation triggering rules) that specify context conditions and situations where recommendations of items of different types should be automatically offered to the user. Besides, triggering priorities can be specified through the definition of exclusion sets. Users have full power to customize the desired behavior according to their preferences.
(c) *Prototype*. We have developed a prototype for Android devices, called R-Rules, which we have used to illustrate our approach and evaluate its performance. The prototype supports the proposal and contributes to the limited availability of practical experiences concerning the implementation of suitable approaches for mobile CARS. Besides the information provided in this paper, additional details about the prototype, including sample videos, snapshots, pictures and demo versions, can be found on the prototype's website (Ilarri, 2023).
(d) *Experimental evaluation*. We have performed an extensive experimental evaluation that shows the feasibility and performance of the proposal. We have verified that the system behaves well with realistic use case scenarios. On the one hand, we have considered an outdoors scenario where a person travels to another city for leisure, which shows that the approach can be useful in tourism scenarios. On the other hand, we have also considered an indoors scenario where a university student needs to avoid overcrowded study rooms where the CO_2 level is too high, which shows that the proposal can be applied in situations where the air quality indoors could be relevant (like, for example, in a COVID-19 scenario, where measures have to be taken to protect the health of people). The two scenarios jointly show the relevance of the different types of rules that we have defined. Besides, a user survey shows that the proposal is suitable to cover the needs of the users (i.e., the types of rules supported allow expressing the usual circumstances under which users would like to receive recommendations of different types).

The work presented in this paper represents a very significantly extended version of the proposal presented in a previous conference paper (Ilarri et al., 2021), where a preliminary version of our triggering approach was presented. Among the improvements, we can mention a more advanced and refined approach and prototype, which now offer support for the management of new types of triggering rules (that we call server-based rules), an evaluation with two use-case scenarios, a survey with users, more detailed explanations, and diagrams that will facilitate understanding and adapting/extending the proposal. The organization of the rest of this paper is as follows. In Section 2, we focus on related works and highlight the contributions to the state of the art. In Section 3, we introduce the architecture designed to provide rule-based push recommendations on mobile devices and the types of rules considered in our approach. In Section 4, we analyze several existing technological options to develop our approach and present the R-Rules prototype, which offers a friendly user interface that facilitates the triggering of recommendations according to the user preferences. In Section 5, we show an experimental evaluation, including use case scenarios to verify that the system responds well to context changes, a user survey, and a performance evaluation where the latency and scalability of the proposal are analyzed. Then, in Section 6, we summarize our conclusions and outline future work. Finally, two appendices provide complementary information about the prototype and the experimental evaluation.

2. Related work

This work falls within the realm of recommender systems –RS– (Lu et al., 2015; Park et al., 2012; Ricci et al., 2022), which have attracted significant research interest due to their capability to alleviate the information overload of users when they need to choose among a large set of options. More specifically, the focus is on recommender systems for mobile computing scenarios, and particularly mobile context-aware recommender systems –CARS– (Adomavicius et al., 2021, 2011; Adomavicius & Tuzhilin, 2005, 2008; del Carmen Rodríguez-Hernández & Ilarri, 2021; Colombo-Mendoza et al., 2017; Liu et al., 2013; Pimenidis et al., 2018), which exploit dynamic context information (such as the location of the user, the time of the day, weather information, etc.) in order to provide more appropriate recommendations. The research literature in this area is very extensive, and therefore we do not intend to provide here a full review of existing approaches; a recent survey on mobile context-aware recommender systems using artificial intelligent techniques can be found in del Carmen Rodríguez-Hernández and Ilarri (2021) and a systematic review on the use of machine learning algorithms in recommender systems in Portugal et al. (2018). Besides, some works, such as Livne et al. (2022), have focused on reducing the dimensionality of the *User x Item x Context* utility matrix in the context of CARS, as the number of contextual variables could be high; an overview of other prior dimensionality reduction techniques used in context-aware recommender systems can be found in Raza and Ding (2019). Instead, we will explain how prior research literature justifies and motivates the work presented in this paper and emphasize the novel aspects of our proposal against the current state of the art.

Push-based (or proactive) recommender systems are usually considered more appropriate than pull-based (or reactive) recommender systems (del Carmen Rodríguez-Hernández & Ilarri, 2021; Sabic, 2016), as they minimize the effort from users, which is particularly relevant in mobile computing scenarios, where proactive RS are able to offer suggestions to mobile users without requiring an explicit request using a mobile device. Besides, according to Zhang and Sundar (2019), users perceive a higher quality in the recommendations received when they are provided proactively, and therefore a reactive approach would compromise the user experience and satisfaction. However, push-based recommendations can also elicit higher privacy concerns (Zhang & Sundar, 2019). Based on the current state of the art, in this work, we have proposed an approach that is suitable for scenarios of proactive recommendations but that at the same time protects the privacy of users. As opposed to other proposals, users have control over the data they want to share, the need to communicate context data to a server to decide whether specific types of recommendations should be offered is completely avoided (as the decision is taken locally on the mobile device), and users can flexibly customize a proactive behavior adapted to their preferences.

With this work, we contribute to bridging the gap between CARS and mobile computing (del Carmen Rodríguez-Hernández & Ilarri, 2021), reinforcing a more active role of mobile devices beyond just being simple clients. Our proposal includes a generic architecture that can accommodate context-aware recommender systems for a variety of use case scenarios. Although there has been a significant amount of research in context-aware computing (Baldauf et al., 2007; Tanter et al., 2006) and also in the field of CARS, practical experiences on the implementation of suitable approaches for CARS are rather limited. Besides our previous conference paper (Ilarri et al., 2021), significantly extended by this one, the most related work is our previous proposal (Ilarri & Herrero, 2018), based on the theoretical foundations described in Hermoso et al. (2015), but that work was focused on the underlying infrastructure (not on the mobile device's architecture) and did not address the recommendation triggering phase and interaction aspects related to the definition of context and triggering rules. Besides, in Hermoso et al. (2018), we outlined potential use cases and provided some examples of context attributes and rules (with SWRL-like syntax) that could trigger diverse recommendation types, but we did not undertake any implementation or testing. So, this paper represents a significant novel contribution over our previous works and existing research.

With the approach described in this paper, we give the user control over the data he/she shares with external servers, ensuring privacy protection, and allow him/her to define and personalize suitable context rules and recommendation triggering rules. In Sharma and Kaur (2015), the idea of utilizing user-defined rules to activate recommendations was also presented (e.g., "If day is Monday then I would like to go to a Restaurant which is closer than 5 KM" is an example of rule shown). In that work, the rules are specified by the user through a web form and translated into RuleML rules (Boley, 2006; Boley et al., 2010); however, only a model is presented and there is no evidence of a working implementation or prototype. A different approach to support proactive recommendations was adopted in Sharma and Kaur (2015), that proposes automatically learning a personal classification model for each user that will predict, given specific context data, whether the timing is suitable to provide recommendations to the user. However, this model for timing recommendations focuses on the recommendation of Points of Interest (POIs) and does not distinguish among different types of recommendations; besides, the users cannot explicitly adapt the proactive behavior to their preferences by defining suitable rules.

User-defined rules have also been exploited in other fields beyond recommender systems. For instance, in the area of network security, rules in SWRL (Horrocks et al., 2004) were defined to enhance the reasoning ability of the model and compensate for the limited description ability of an ontology (Xu et al., 2017). Another example that we can mention is *SECE –Sense Everything, Control Everything–* (Beltran et al., 2011; Boyaci et al., 2011), which is a rule-based context-aware system that uses a natural language-like formal language to define rules for composing different services based on events. However, unlike our work, SECE is a web service and does not run on a mobile device. Therefore, in SECE, all the context data need to be sent to the centralized server, which leads to disadvantages concerning the user's privacy and the amount of wireless communications from the user's mobile device; besides, the definition of rules in our approach is based on easy-to-configure graphical screens, which are more suitable for interacting with mobile devices. As a final example, *LLA –Long-Life Application–* (Karchoud et al., 2017, 2019) is a single, context-aware, distributed mobile app designed for everyday users. The idea is to match services with relevant *situations* (context descriptions) to respond to contextual changes when appropriate; situations can be "injected" by the user, by external providers (such as governments, business and private companies, as well as institutions, organizations, and associations), or by a component of LLA that monitors the user's social environment (e.g., tasks defined in Google Calendar) to suggest possible situations. The motivating idea behind LLA is to make an app go to the user when it is suitable, even if the user does not activate the app. Neither SECE nor LLA belong to the field of recommender systems, whose final goal is to recommend items/activities to users based on their preferences (e.g., previous ratings), and the user's privacy is not a significant concern for them. While proposing user-defined rules is not something new, as those previous works illustrate, our proposal concerns about user-defined rules in the specific context of proactive mobile recommender systems; therefore, we have evaluated that the set of rules proposed is suitable for that context and designed suitable graphical user interfaces to facilitate the definition of the rules using mobile devices.

Other examples of rule-based end-user applications include *IFTTT –If This, Then That–* (IFTTT Inc, 2011–2023; Ovadia, 2014) and the *Shortcuts* app (Apple Inc, 2018–2023). On the one hand, IFTTT is a web-based service that allows users to automate tasks by connecting apps, such as automatically publishing a picture on Twitter after it has been posted on Facebook. Users can define these actions through the IFTTT website or the IFTTT mobile app, which is available for Android and iOS, and currently works with over 700 apps and devices (according to the information shown on its web page). On the other hand, the Shortcuts app, available only for iOS devices, enables users to define conditions for running specific tasks (shortcuts) automatically, such as based on the time of day or the location. Four types of triggers can be considered: event triggers (a designated time of the day, the activation of an alarm on the user's phone, or the completion of a workout on the user's Apple Watch), travel triggers (when the user arrives at or departs from a location, when he/she connects or disconnects CarPlay (Apple Inc, 2016–2023), or when the user's commute is about to start), communication triggers (when receiving specific types of emails or messages), and setting triggers (such as the activation of the airplane mode, changes in the battery level, connecting to a WiFi network or to a selected Bluetooth device, starting or stopping charging, etc.).

The idea of keeping data locally to process them without sending them to a central server, in order to protect privacy and/or save bandwidth, is not new, and has been applied in fields such as health care and computational advertising (Dave & Varma, 2014). For example, a proposal where health data are kept in their original location and analytical tasks are brought to the data provider for local processing was presented in Welten et al. (2022). As another example, filtering data on the edge to minimize bandwidth communications has been proposed in works such as Akrivopoulos et al. (2017). In that work, an illustrating prototype that analyzes Electrocardiogram (ECG) signals and generates alerts locally on the user's mobile device is also presented; in the literature, we can find a plethora of works related with the concept of self-monitoring patients (Lupton, 2013). As a final example, a client-side ads-filtering mechanism to guarantee the privacy of personal data is presented in Carrara et al. (2013). It is also relevant to mention the extensive amount of research that is being performed in the field of federated learning, whose goal is to build machine learning models across distributed devices without exchanging raw data, in order to protect sensitive data; as examples of the work being performed in this area, the interested reader is referred to Liu et al. (2021) and Yin et al. (2021). While the idea of processing data locally on a mobile device is not new, the specific approach presented in this paper is novel; as far as we know, there is no other proposal that allows mobile users of push-based recommender systems to keep complete control of the data they share with other servers/devices. Besides, we have not found any reported experience concerning the development of a prototype that embeds a full rule-based engine middleware in a mobile device.

There are also works that focus on CEP (Complex Event Processing) engines. For example, an engine able to process complex event queries with time windows and partition-by event correlation operators was presented in Bucchi et al. (2022). Works related to CEP are complementary to ours, as we do not aim at developing a CEP engine but at defining a suitable and customizable approach that uses a CEP engine executed in a mobile device to trigger recommendations

when appropriate. Whereas there are several surveys on CEP technologies (Dayarathna & Perera, 2018; Giatrakos et al., 2019), they do not focus on their use in mobile devices to create efficient and proactive recommender systems.

Thus, to the best of our knowledge, existing works primarily consider the deployment of CEP engines on fixed servers rather than on mobile devices. However, executing the recommendation triggering phase on the mobile device rather than on a server brings relevant benefits: it reduces wireless communication from the device (communications of context updates), and thus lower the energy consumption; besides, it contributes to preserving the user's privacy, which is a key aspect when dealing with proactive recommender systems (Zhang & Sundar, 2019). Moreover, as keeping the context data continuously updated in an external server would be challenging, approaches where the triggering decision is taken on a fixed server would require an efficient communication policy between the mobile device and the remote CEP engine, or a policy for deciding which context variable changes should be communicated to the server (since they could trigger a recommendation process). Alternatively, appropriate context variable update policies (such as dead-reckoning policies or the use of prediction functions) could be applied. These types of policies have been previously studied for location data (Wolfson et al., 1998, 1999); for example, dead-reckoning policies are used to send an update of the current location when the difference between the actual location and the location that can be predicted based on the last communicated location and speed exceeds a certain threshold, in order to minimize the amount of data transmitted between the mobile device and the server. Similar policies have been applied for general sensor data; for example, the architecture presented in Ilarri et al. (2009) exploits prediction functions to minimize the communication and processing overhead.

Summing up, the main novelty of this work lies in the proposal of a complete solution to trigger suitable recommendations for mobile users while preserving their privacy. For that purpose, the context data are processed locally on the mobile device in order to decide when a recommendation of a specific type should be fired, which also allows reducing the amount of wireless communications with external servers and so increasing the battery's lifetime. Besides, we have analyzed which types of rules are relevant in the context of mobile recommendations and developed a prototype that illustrates the proposal. This prototype facilitates the interaction of mobile users with proactive recommendation servers and has been validated experimentally. Up to the authors' knowledge, besides our work, there is no other reported experience concerning the development of a prototype that embeds a full rule-based engine in a mobile device to trigger the recommendations.

3. Architecture of the system and types of rules

In this section, we present a high-level overview of the approach proposed to deploy recommender systems that provide proactive (push-based) recommendations to mobile users when appropriate (Section 3.1), without requiring any user intervention, as well as the types of rules managed in the system (Section 3.2).

3.1. Device-based triggering recommendation approach

The focus of the proposed architecture is on the detection of circumstances that should trigger a recommendation process through the use of a *Complex Event Processing* (*CEP*) module on the mobile device. The mobile device captures context data from various sensors and sends them to the CEP engine, which decides whether a recommendation should be made and the type of items to recommend. If a recommendation is triggered, the mobile device communicates with an *Environment Manager* (*EM*) server which provides recommendations (i.e., an EM is a recommendation provider). An EM thus offers a recommendation service to the devices that are located within the *environment* managed by that EM, and this proposal supports easily adding new recommendation providers (new EMs) as required. Environments are physical or virtual areas that contain users that can benefit from a specific type of recommendations. For example, in Fig. 1, we show, for illustrative purposes, three different environments: *E1* represents the environment of the Museum of Modern Art in New York, *E2* is the Midtown Manhattan area of New York, and *E3* is the environment for the whole city of New York. Each of these environments is managed by its corresponding EM (EM1, EM2 and EM3, respectively). As shown in the figure, environments can overlap; for example, users *user1* and *user2* are located within the environment *E1* (as they are at the moment visiting the MoMA), but also within the environment *E2* (as they are in Midtown Manhattan) and in *E3* (as they are in New York), so they could potentially receive recommendations from any of these three environments. The EM can interact with multiple mobile devices at the same time (e.g., in the figure, *EM2* can interact with *user3*, *user4*, *user1* and *user2*) and provide a set of recommended items to each of them. The items recommended can be filtered on the mobile device based on the user's private preferences, which are not shared with the EM. The concepts of environment and EM were presented in Hermoso et al. (2015), focused on the external supporting infrastructure and where we did not address the architecture for mobile devices and the triggering of recommendation rules as we do in this paper; thus, we add here a rule definition and evaluation module.

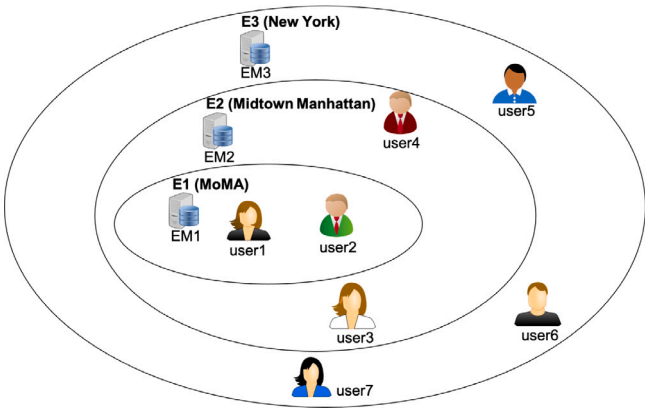

Fig. 1. Example of underlying infrastructure: overlapping environments, each one managed by a server.

Therefore, as shown in Fig. 2, we propose an architecture with two main types of physical components: environment managers (recommendation servers) and mobile devices (recommendation clients). Both play a key role in the recommendation process, as the sensing and processing capabilities of mobile devices are exploited to alleviate the workload on the environment managers and also for privacy purposes. Thus, the architecture is based on three essential principles:

1. *The recommendation triggering phase is executed on the mobile device*. This requires a CEP engine that is capable of running on mobile devices. Running the CEP engine on a fixed server on the network would result in increased wireless communications and negatively affect the battery life of the mobile device, as it would require constant context data updates. Instead, the CEP engine automatically considers new context data with a specific frequency (e.g., every 30 seconds) locally on the mobile device of the user. The data update frequency can be configured according to the existing needs.
2. *The user's privacy is preserved*. The proposed architecture protects user's sensitive data through two complementary mechanisms. On the one hand, the recommendation triggering phase takes place on the mobile device, avoiding the need to send context data to a server. Thus, for example, we can avoid the continuous communication of context information such as the precise

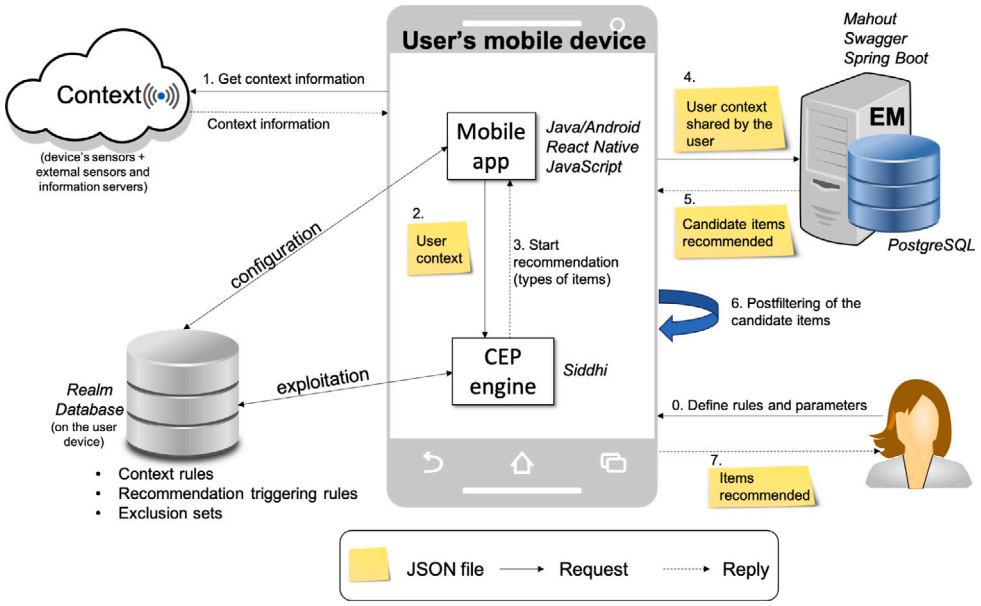

Fig. 2. High-level architecture focused on the triggering of recommendations.

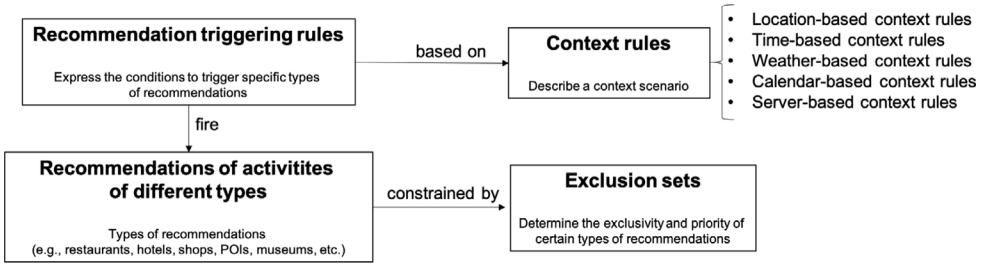

Fig. 3. Types of rules managed in our proposal: context rules and recommendation triggering rules.

location of the user, which may be needed in order to decide that the recommendation of a specific type of items should be triggered. On the other hand, when a recommendation is triggered, the user has control over the data shared with the external recommendation server (i.e., the EM). The EM can use shareable, non-sensitive preferences for prefiltering the candidate items to recommend, but private sensitive user preferences will only be used for postfiltering those items on the mobile device itself. For example, the user may have a dietary restriction (e.g., due to having celiac disease) but may not want to communicate it to a server, as it represents potentially-sensitive health data; nevertheless, this information could be used locally in the mobile device to remove recommendations of eating places that usually offer foods rich in gluten (e.g., bakeries).

3. *The user can customize the desired behavior*. The user can define and customize his/her preferences regarding the activation of recommendations of different types. For this purpose, it is possible to define two different types of rules on the mobile device of the user (see Section 3.2): *context rules* (described in Section 3.2.1) and *recommendation triggering rules* (presented in Section 3.2.2). Besides, the user can also define *exclusion sets* to avoid the simultaneous activation of specific types of recommendations at the same time (as explained in Section 3.2.3).

Both context rules and recommendation triggering rules are managed by a CEP engine that runs on the mobile device. The purpose of this engine is to detect events that should activate specific recommendations. To achieve this in an efficient and privacy-preserving way, the CEP engine must be capable of processing all event detection rules on the mobile device.

3.2. Types of rules

In this section, we describe the types of rules managed in our proposal, which are summarized in Fig. 3. First, in Section 3.2.1, we present context rules. Then, in Section 3.2.2, we describe recommendation triggering rules. Finally, in Section 3.2.3, we explain the use of exclusion sets.

3.2.1. Context rules

Context rules are rules that define context conditions or situations that may be of interest to the user. We currently consider the following types of context rules: location-based context rules, time-based context rules, weather-based context rules, calendar-based context rules, and server-based context rules. Several snapshots of our prototype showing examples of the different types of context rules can be found in Fig. 4 and Fig. 5. In the following, we describe the different types of context rules considered in our approach.

Spatio-temporal criteria can be set through location-based, time-based and calendar-based context rules. On the one hand, *location-based context rules* are activated when a specified location-dependent constraint is satisfied. For instance, the user could set a location-based context rule called *InMillenniumPark*, which is satisfied when the user is in the Millennium Park area in Chicago (Illinois, United States), that is, around a given GPS location (<41.882702, −87.619392>); another example is a location-based context rule *AtHome*, which is true when the user is at his/her home. On the other hand, *time-based context rules* are activated when a specified time constraint is satisfied. Examples of time-based context rules could include *TimeToWakeUp*, indicating the time interval when the user plans to wake up, or *LunchTime*, which

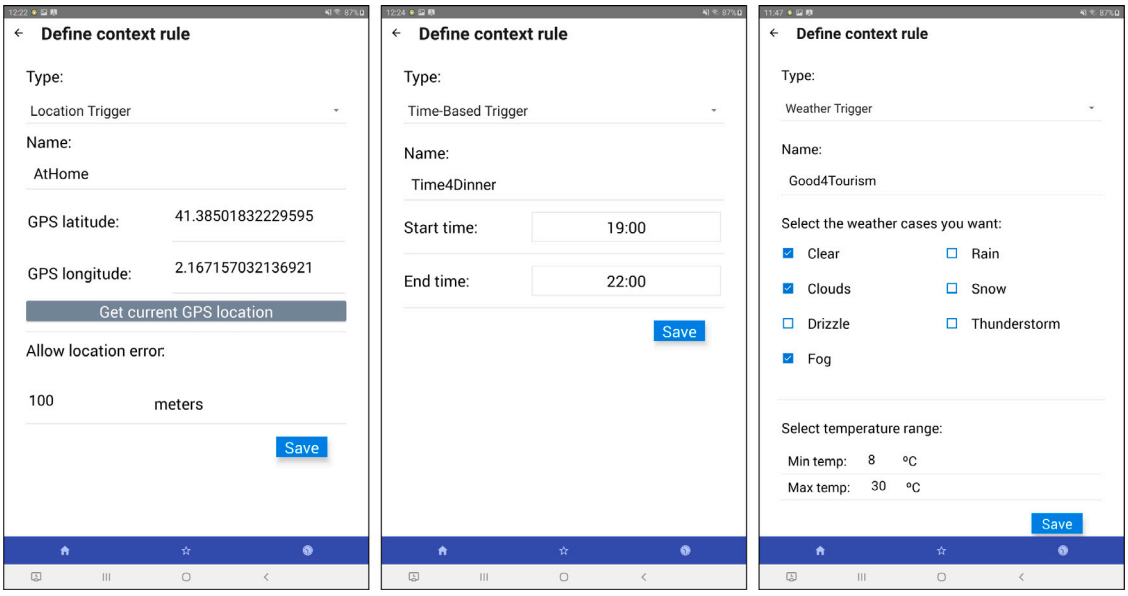

Fig. 4. Screens to define context rules (1/2): location-based (left), time-based (middle), and weather-based (right).

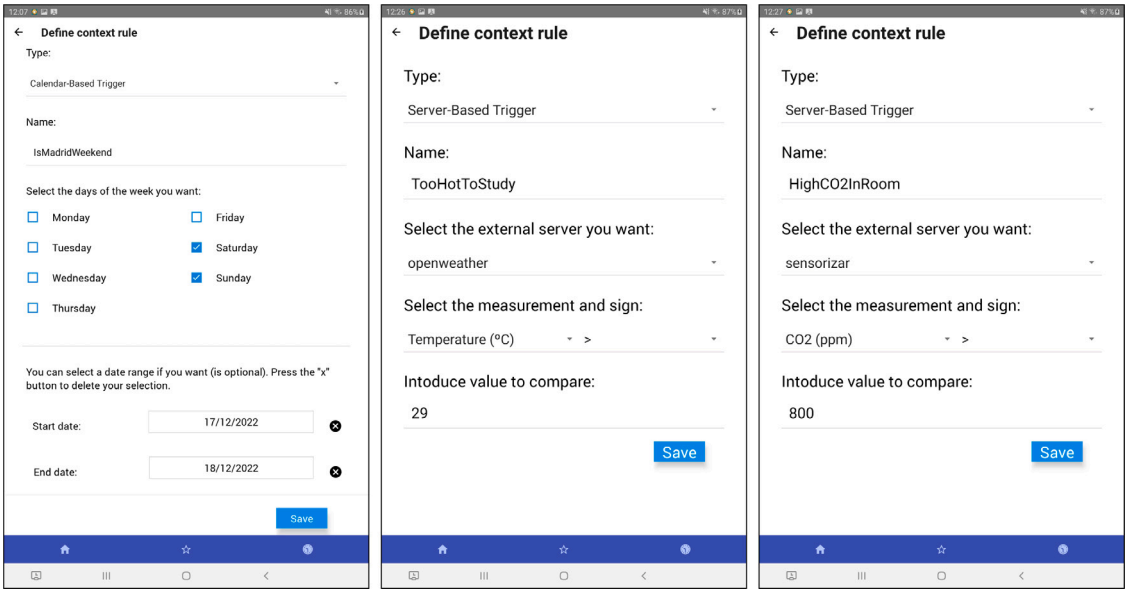

Fig. 5. Screens to define context rules (2/2): calendar-based (left), server-based with OpenWeather (middle), and server-based with sensoriZAR (right).

could depend on the user's preferences and the region's habits where the user is located. Besides, *calendar-based context rules* are activated when the specified calendar conditions are satisfied. For example, the user could configure a calendar-based context rule *VeniceVisit* that is defined using the dates when he/she will be visiting Venice. As another example, he/she could set up a rule *IsWeekend* that is satisfied when the day of the week is Saturday or Sunday; alternatively, a person working on Saturday could define the *IsWeekend* context rule considering only Sunday as part of the weekend.

Criteria concerning other context data, such as the weather or sensor data accessible through a web service, can be set by using weather-based and server-based context rules. *Weather-based context rules* are activated when the specified weather conditions are satisfied. For example, we could have a user-defined weather-based context rule *Good4Walking* that is satisfied when the sky is clear and the temperature is within a range that the user considers suitable to perform that kind of activity; similarly, a user could define a weather-based context rule *Good4Cinema* that is activated when it is raining or the weather is not good to perform other types of activities outdoors. On the other hand, *server-based context rules* are activated when a specific value provided by an external server satisfies the condition specified (i.e., when the value provided by the server is equal, smaller than, or higher than a specific value indicated by the user). For example, the user could define a server-based context rule *PoorRoomAirQuality* that is activated when the CO_2 of the room where the user is located (provided by an external server) exceeds a given threshold. Similarly, the user could set up a context rule *TooHot* that is activated when the temperature of the room is too high according to his/her temperature preferences.

The types of context rules considered (location-based context rules, time-based context rules, weather-based context rules, calendar-based context rules, and server-based context rules) correspond to different types of contextual information frequently used in the literature (the location, the time, the weather, the calendar and other context variables that can be queried through a server). In our work, we put the

emphasis on the functionality provided to the end user. For example, through server-based context rules it is possible to consider a wide variety of context data (such as the CO_2 level, the amount of traffic, or the crowdedness of an area, as long as this contextual information can be accessed from the mobile device by querying a service), a calendar-based context rule supports conditions on the day of the week and also on specific date ranges, and a weather-based context rule allows setting conditions for different types of weather variables.

3.2.2. Recommendation triggering rules

Recommendation triggering rules (or, for brevity, just *triggering rules*) are rules that determine when a specific type of recommendation should be automatically activated. They are defined by specifying the context rules that must be satisfied (an *AND* operator is considered to join the context rules indicated, and each context rule can also be negated by using a *NOT* operator). For instance, a user could define a rule *LunchOutside* that turns on the recommendation for restaurants only if it is lunch time and the user is not at home (combination of a time-based context rule and a location-based context rule).

Multiple recommendation triggering rules could activate the same type of recommendation; for example, the user could define two rules, *LunchOutside* and *DinnerOutside*, to activate a recommendation of restaurants for lunch time and also for dinner time. Of course, the user can also define multiple recommendation triggering rules for different types of recommendations; for example, he/she could define a recommendation triggering rule to recommend restaurants and another one to recommend POIs.

3.2.3. Exclusion sets

Context rules can overlap, in the sense that the same condition (e.g., the same time interval in the case of a time-based context rule) could be used in several context rules. We could also have a certain condition (e.g., a condition specifying that the temperature is above 29 °C) in a context rule and the opposite condition (i.e., a condition indicating that the temperature is below 29 °C) in another context rule. Similarly, the same context rule (e.g., a context rule specifying that the user is not at home) can be part of more than one recommendation triggering rule, and we could also have a context rule and its negated version (e.g., a context rule expressing that the user is not at home and another one indicating that he/she is at home) as part of different recommendation triggering rules.

However, by default, there are no real conflicts between recommendation triggering rules. It may occur that several triggering rules are satisfied at the same time; this is not interpreted as a conflict, but as the need to trigger different types of recommendations at the same time. If this is not what the user wants, then the user can set up *exclusion sets* to avoid receiving certain types of recommendations simultaneously and specify their priority within the set. For example, the user could set up an exclusion set called *FirstEatThenWatch* that assigns a higher priority to the recommendation of restaurants over the recommendation of cinemas, in case both types of recommendations could be activated at the same time.

In Fig. 7 (located in Section 4), we can see other examples of exclusion sets using our prototype. Specifically, the exclusion set *MuseumsBetter*, shown in the middle of the figure, prioritizes museums over POIs and shops, and *ShowsBetter*, shown on the right, sets a preference for shows over restaurants.

4. R-rules prototype

In this section, we present our prototype, that we call *R-Rules*, emphasizing the key elements needed to support the detection of rules to activate suitable recommendations. First, in Section 4.1 we summarize the technologies considered for the development of the prototype. Then, in Section 4.2, we present the mobile app and its user interface. Besides, we provide additional information about R-Rules in Appendix A.

4.1. Technologies considered

Concerning potential development platforms, we have followed an approach targeting a hybrid mobile app using React Native (Eisenman, 2015), as we explain in more detail in Appendix A.1. In this section, we focus on the technological approach that can be adopted to detect when the recommendation of a specific type of item should be activated. For this, CEP systems (Bucchi et al., 2022; Cugola & Margara, 2012; Dayarathna & Perera, 2018; Giatrakos et al., 2019) provide the capability of defining rules for pattern detection, including complex event patterns through aggregation and composition of events. Our aim is to incorporate this type of systems into our approach to determine if a given context is suitable for triggering the recommendation of a particular type of items, based on the establishment of appropriate context rules.

In Appendix A.2, we provide some details about different technological alternatives that we have considered for the detection of suitable contexts: Esper and Asper, Siddhi, Apache Flink, Drools, and Microsoft StreamInsight. Our main objective is to find a CEP engine that can run on mobile devices (particularly, on Android devices, due to their popularity), which is only fulfilled by Asper and Siddhi (see Table 1). As Siddhi is more up-to-date, we chose Siddhi as the technological solution to implement the triggering decision module (more details can be found in Appendix A.2.6). Other CEP technologies can only be used if a different architecture where the rule detection takes place on an external server (the EM) is adopted.

4.2. Interaction with the CEP engine and mobile app

As our main focus is the problem of triggering mobile recommendations, in this section we mainly describe the integration of the Siddhi CEP engine and highlight its functionalities related to defining and customizing user rules. Our solution uses React Native (Eisenman, 2015) with the main code of the mobile app written in JavaScript; additional details about React Native can be found in its website at https://reactnative.dev. The Siddhi engine can handle events from various sources and process them as needed by the application, and then deliver results via various channels like HTTP, Kafka, or email. In our implementation, we exploit the Siddhi library to transmit and receive events using Java.

We integrated Siddhi into our Android prototype as a service of Android, that is, as a *Service* application component (https://developer.android.com/guide/components/services), allowing it to run background operations without impacting the mobile app's Graphical User Interface (GUI). Our implementation utilizes a bound service, which provides a client/server interface for interaction with the service. Bound services are one of the three types of services available in Android, along with foreground and background services.

In Appendix A.3, details about the packages composing the R-Rules prototype and about the communication with Siddhi are provided. It should be noted that Siddhi reports only the activation of recommendation triggering rules. When a recommendation triggering rule is activated (i.e., when the current context matches the conditions defined in the rule), an event is stored in a *Result* stream and retrieved using a *callback* (https://reactnative.dev/docs/native-modules-android), as documented in the official documentation of Siddhi (https://siddhi.io/en/v5.1/docs/siddhi-as-a-java-library). Instead of just considering the type of recommendation associated with the rule fired, the mobile app collects all the types of recommendations corresponding to the rules fired within a specified batch time window (e.g., 5 seconds in the current prototype). The resulting batched output is processed to determine which recommendations should be initiated based on user-defined exclusion sets and the priority of types of recommendations within each set, as explained in Section 3.2.3. For example, <contextId, {restaurantRecommendation, museumRecommendation, parkRecommendation}> could be a resulting batched

Table 1
Summary of technologies considered for complex event processing.

Technology	Open source	Executable on Android	Last update mentioned	Main goal
Esper	✓	–	May 2023 (version 8.9.0)	Complex Event Processing (CEP)
Asper	✓	✓	March 2013 (based on Esper 4.8.0)	Complex Event Processing (CEP)
Siddhi	✓	✓	February 2023 (version 5.1.27)	Complex Event Processing (CEP)
Apache Flink	✓	–	May 2023 (version 1.17.1)	Processing of data streams
Drools	✓	–	September 2023 (version 9.44.0)	Business Rules Management System (BRMS)
Microsoft StreamInsight	–	–	June 2014 (version 2.3, part of SQL Server 2014)	Complex Event Processing (CEP)

output, the recommendation of restaurants and museums could be part of the same exclusion set (which means that only one of them can be active at the same time), and the recommendation of museums could have a higher priority (so it will be the preferred option between those two exclusive types of recommendations). This batching is achieved using the *timeBatch* function provided by Siddhi (https://siddhi.io/en/v5.1/docs/query-guide/#window), which supports the concept of *batch windows*.

Concerning the graphical user interface of the mobile app, the context rules' conditions can be customized by the user as shown in Fig. 4 and Fig. 5 (located in Section 3.2.1). When a location-based triggering rule is created or edited, besides entering a GPS location by hand, it is also possible to click on a "Get current GPS location" button, which could provide a quite precise location when outdoors or just an estimated location value (or simply the last known location) when the user is indoors (using positioning methods other than GPS, such as information about the WiFi network the device is connected to), as illustrated on the left of Fig. 4. Among the different types of context rules, server-based context rules provide high flexibility, as they support conditions based on information provided by any external service. Currently, in our prototype, the user can select one of these two servers: *OpenWeather* (https://openweathermap.org) and *sensoriZAR* (Smart Cities Lab, 2021–2023), as shown in the middle and on the right of Fig. 5. sensoriZAR is an IoT platform for monitoring buildings, developed by the Smart Cities Lab of the Aragon Institute of Engineering Research (https://smartcities.unizar.es), that provides data from sensors deployed in different buildings at the University of Zaragoza (Spain). OpenWeather is also used for the implementation of weather-based context rules. The prototype interacts with these servers by using the corresponding API (Application Programming Interface). Through sensoriZAR we can obtain, for example, information concerning the CO_2 level and the room temperature in different buildings of the University of Zaragoza. Additional servers can be added by defining basic data about the servers and the needed server requests in a configuration file.

The left of Fig. 6 shows the screen that allows the management of recommendation triggering rules (definition, modification, deletion, and temporary deactivation of the triggering rules). In the middle of Fig. 6, an example of a user-defined recommendation triggering rule is shown: the recommendation of shows will be activated if the user is at home and it is not lunch time, dinner time or the user's resting time. On the right of Fig. 6, we can see the screen for managing context rules (definition, modification, and removal of context rules).

In Fig. 7, we show the screens that allow defining exclusion sets (sets of types of recommendations that should not be activated at the same time, as explained in Section 3.2.3) and their priorities within the set. Thus, in the example shown on the right of Fig. 7, the user indicates that when the recommendations of shows and restaurants could be activated at the same time only one of them should actually be activated according to the order shown (in this case, the recommendation of shows has the highest preference).

Finally, in Fig. 8, we show some other screens of our current prototype, that allow the user to set up the required environment. For example, on the right of Fig. 8 we can see how the user can select which data he/she is going to share with the Environment Managers, according to his/her privacy concerns. For more details about the prototype, we refer the interested reader to the web site of R-Rules (Ilarri, 2023), which includes sample videos, snapshots, pictures and a version of the prototype.

5. Experimental evaluation

In this section, we describe the experimental evaluation that we have performed (see Table 2 for an overview). First, in Section 5.1, we present two scenarios (one outdoors and another one indoors) that we have considered to verify that the right types of recommendations get activated according to the context changes, thus showing that the system works well. Then, in Section 5.2, we present a user survey where we have collected information about user requirements in order to evaluate if our approach can suitably cover those user needs. Afterwards, in Section 5.3, we evaluate the performance of our proposal. Finally, in Section 5.4, we summarize some conclusions of the performance evaluation. It should be noted that, whereas we evaluate the quality of the types of recommendations triggered (in Section 5.1), we do not evaluate the quality of the specific items recommended because, in this paper, we do not propose any specific context-aware recommendation algorithm; instead, once the types of recommendations that should be activated are detected by using our approach, any CARS could be used to provide suitable recommendations of the relevant types of items. Some additional details about the evaluation are provided in Appendix B.

5.1. Triggering evaluation

In this section, we describe how we have verified, using test scenarios, that the right types of recommendations are fired according to the existing contextual attributes. In Section 5.1.1, we consider an outdoors scenario, and in Section 5.1.2 we focus on an indoors scenario.

5.1.1. Triggering evaluation in an outdoors scenario

Initially, we focus on an outdoors scenario that illustrates the use of context rules such as weather-based context rules, location-based context rules, time-based context rules, and calendar-based context rules.

Description of the outdoors scenario

Let us imagine a user called Alice who is visiting Madrid for tourist purposes during the 16, 17 and 18 of December 2022. She has already visited Madrid before and she stays in an apartment in the Malasaña area and mainly moving around nearby areas (Chueca, the area of the Royal Palace of Madrid, etc.), although she is willing to take public

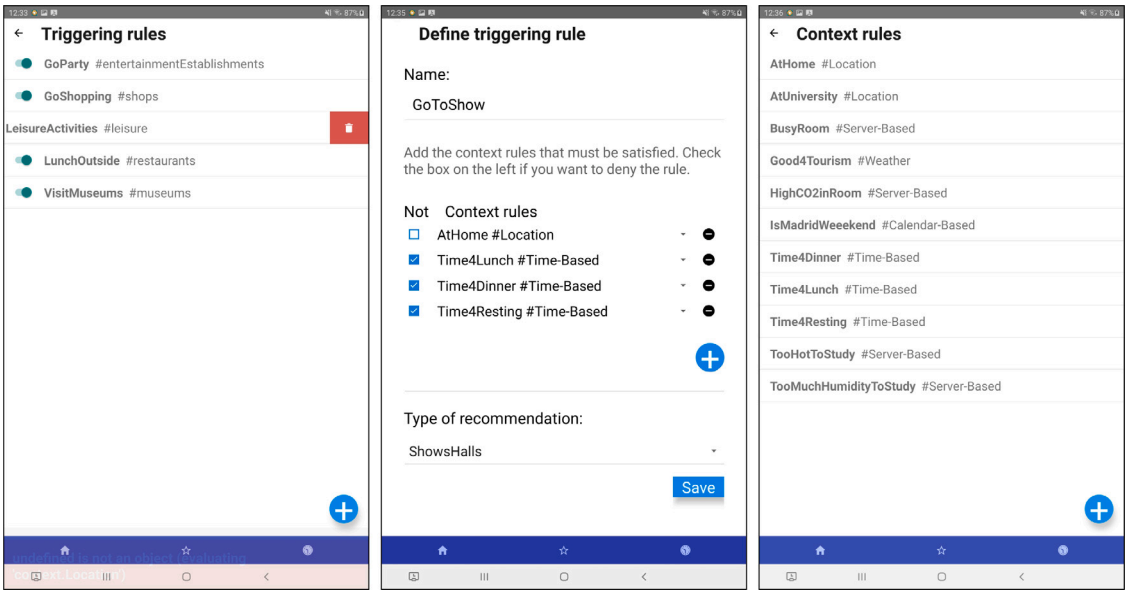

Fig. 6. Screens to manage triggering rules (left), to define a new triggering rule (middle), and to manage context rules (right).

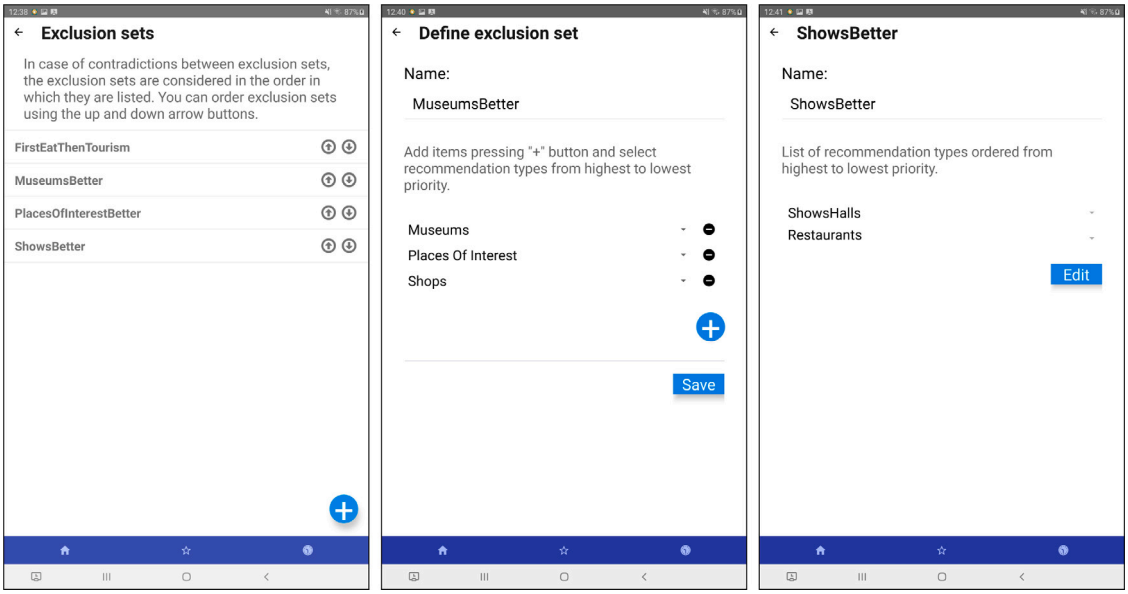

Fig. 7. Screens to manage exclusion sets (left), to define an exclusion set (middle), and to edit an existing exclusion set (right).

transportation to go to more distant places in the city. She is interested in different types of recommendations except lodging (as she already has a hotel). She defined 11 context rules, 7 triggering rules (each activating a different type of recommendations) and 3 exclusion sets.

More specifically, to identify situations when she would like to receive recommendations of different types of items, she defined the context rules indicated in Table 3. Based on these context rules, she also defined the recommendation triggering rules shown in Table 4; some of these triggering rules, like *LeisureActivities*, were deactivated by Alice at some point during her visit to Madrid, and besides, on Sunday she edited the *VisitMuseums* triggering rule to remove its dependence on the context rule *Good4Museums*. Finally, Alice defined several exclusion sets, to avoid receiving recommendations of items of certain types at the same time: *FirstEatThenTourism* prioritizes the recommendations of restaurants over points of interest, *MuseumsBetter* prioritizes museums (then points of interest, and finally shops), *PlacesOfInterestBetter* prioritizes places of interest over leisure activities, and *ShowsBetter* (exclusion set not initially defined by Alice, but activated by Alice on Saturday, December 17, 2022) gives preference to shows rather than other leisure activities.

To simulate context changes, we defined a set of user contexts (compiled in a JSON file); every 30 seconds, a new context was read and sent to Siddhi. In this way, we could easily force the context changes that we wanted to emulate for testing purposes.

Evaluation in the outdoors use case scenario

Fig. 9 represents an execution timeline that shows how the detection of the types of recommendations to fire and the exclusion sets work properly. The consecutive numbers 1, 2, 3, etc., represent a context identifier; the labels on top indicate the context changes that are relevant and actions performed by Alice regarding the activation and deactivation of triggering rules; the bottom labels linked to the contexts by an arrow represent the types of recommendations that Siddhi has detected as relevant, underlining the ones that are actually

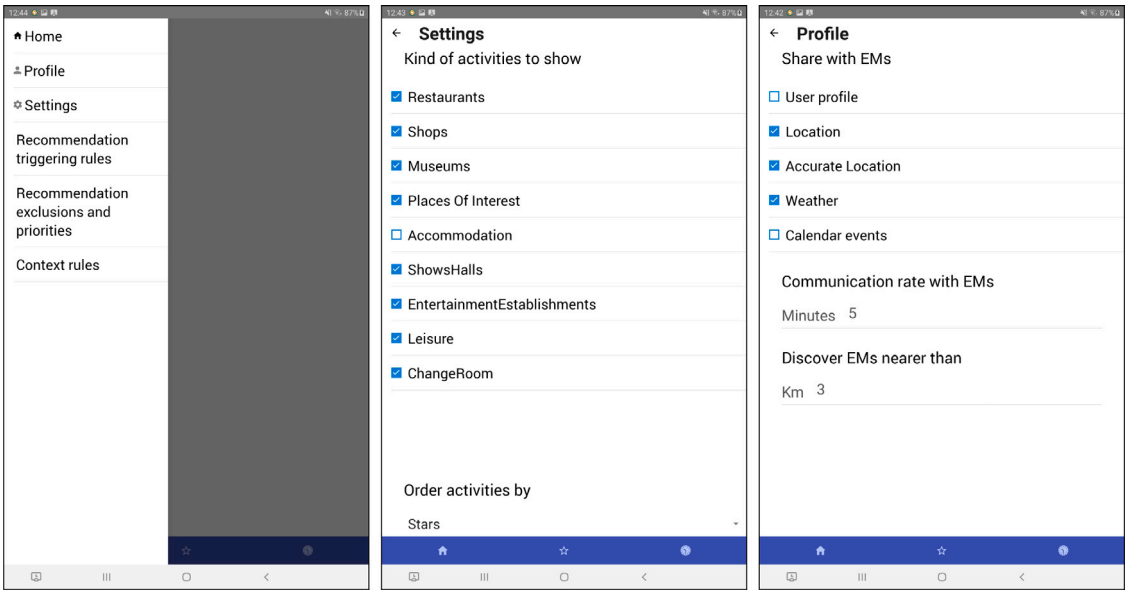

Fig. 8. Screens to set up the application: structure of menus (left), overall settings (middle), and data to share with EMs (right).

Table 2
Overview of the experimental evaluation.

Experiments	Section	Goal	Focus	Datasets
Triggering evaluation	5.1	Verify that the right types of recommendations are fired according to the existing contextual attributes, in different types of scenarios: -Outdoor scenarios (Section 5.1.1). -Indoor scenarios (Section 5.1.2).	Detection of potential failures. There are two types of possible failures: -Recommendations that were activated when they should not. -Recommendations that were not activated when they should.	-Context rules, triggering rules and exclusion sets defined for the experiments to represent the required preferences of the user. -Context changes defined for the experiments to represent the timeline required. -In the indoor scenario, context data about buildings provided by the sensoriZAR server (monitoring platform). -In the outdoor scenario, POIs obtained from Madrid's Open Data Portal.
Evaluation of the suitability of the types of rules supported	5.2	Verify that the current support of context rules and recommendation triggering rules is suitable to satisfy most user requirements.	Identification of users' needs that might not be covered with the current approach.	-Survey filled by users.
Performance evaluation	5.3	Assess the performance of the recommendation triggering phase: -Triggering latency (average, minimum, maximum). -Scalability as the number of recommendation triggering rules increases.	Detection of potential situations that might overload the system.	-Consideration of 7 context variables to build context rules. -Random combination of context rules to generate recommendation triggering rules. -Random generation of example contexts.

activated according to the exclusion sets defined (i.e., the output of the recommendation triggering module). For clarity, we briefly discuss some aspects of the figure:

- When the user is in context 4, according to the rules defined, the types of recommendations that could be activated are *PlacesOfInterest* and *Restaurants*. However, the exclusion sets defined in this case lead to the recommendation of restaurants (there is an exclusion set, called *FirstEatThenTourism*, that prohibits the activation of both types of recommendations at the same time and gives more priority to the recommendation of restaurants).
- In contexts 5, 14, 15 and 17, no recommendation should be provided.
- In context 2, the types of recommendations activated are *Leisure* and *Museums*. In context 3, there is no relevant context attribute change that could modify the types of recommendations to activate; however, as Alice deactivates the triggering rule that fires the recommendation of items related to *Leisure*, only the recommendation of museums is activated.
- Before processing context 17, Alice modifies the triggering rule that activates the recommendation of museums (*VisitMuseums*), by removing the associated weather-based context rule. This leads to the recommendation of museums even in that context, where the sky is cloudy but it is not raining.
- Before processing context 9, Alice adds a new exclusion set that assigns a higher priority to shows than other leisure activities.

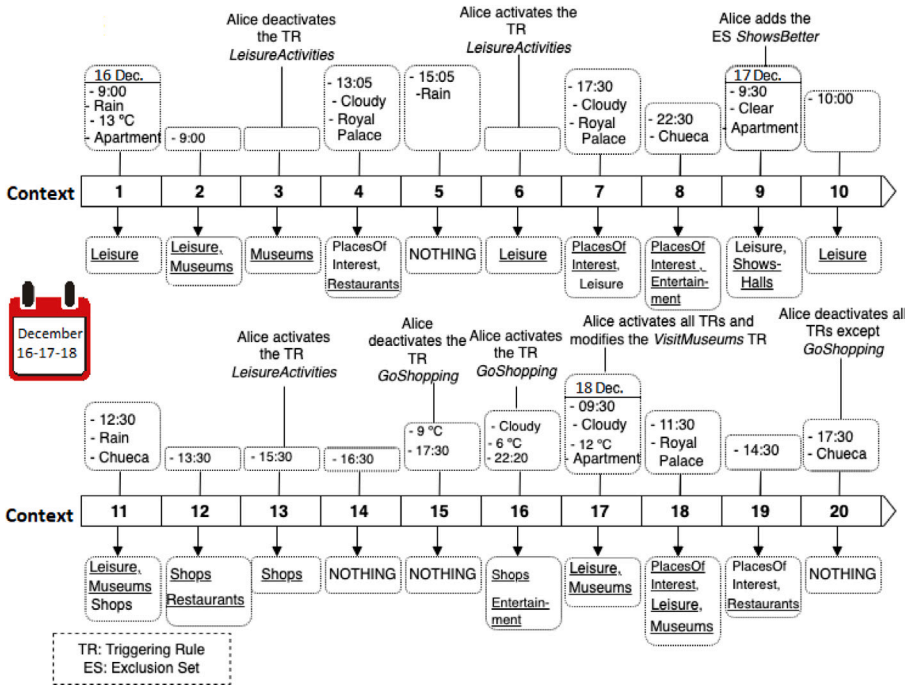

Fig. 9. Timeline for the example outdoors scenario: contexts and types of recommendations activated.

Table 3
Context rules defined by Alice in the outdoors use case scenario.

Name	Type	Description
AtApartment	location-based	Coordinates of Alice's apartment in Madrid
AtChueca	location-based	Coordinates of the Chueca neighborhood
AtPalacioReal	location-based	Coordinates of the Royal Palace of Madrid
Time4Lunch	time-based	Alice's lunch time is between 13:00 and 15:00
Time4Dinner	time-based	Alice's dinner time is between 21:00 and 22:30
Time4Party	time-based	Alice's usual party time is between 22:00 and 00:00
InTheMorning	time-based	Alice's morning is between 9:30 and 14:00
IsMadridWeekend	calendar-based	The 17th and 18th of December 2022 (Saturday and Sunday)
ShowDay	calendar-based	Alice's top day for shows is Saturday
Good4Tourism	weather-based	For Alice, doing tourism is OK if it is clear, cloudy or foggy and the temperature is between 8 °C and 30 °C
Good4Museums	weather-based	Alice prefers visiting museums if it rains or there is storm and the temperature is between 0 °C and 15 °C

As a consequence, in context 9, only the recommendation of *ShowsHalls* is activated.

All the results shown in Fig. 9 are correct. This experiment, as well as other tests performed in similar scenarios, show that the detection approach works correctly: the appropriate types of recommendations are triggered depending on the context and the exclusion sets defined. Besides, the experiment illustrates how the approach can be applied in real outdoor scenarios.

5.1.2. Triggering evaluation in an indoors scenario

Now, we consider an indoors scenario that illustrates the interest of using server-based context rules. This scenario also shows the usefulness of the approach in a situation where the air quality indoors could be relevant, for example to avoid the propagation of virus like the COVID-19.

Description of the indoors scenario

After her weekend in Madrid, Alice is now at the School of Engineering and Architecture at the University of Zaragoza, as she is pursuing an engineering degree there. She is located in a building in the campus *Río Ebro*, but she does not like rooms with a too-high temperature or humidity, as this can affect her concentration while studying. Besides, she is aware of the importance of air quality (in particular, the relevance of suitable CO_2 levels), for example to avoid the propagation of respiratory infections (Agarwal et al., 2021; Peng & Jimenez, 2021). Taking this into account, she defined the context-rules indicated in Table 5. Based on these context rules, she also defined the recommendation triggering rules shown in Table 6.

With the triggering rules defined, Alice will receive a recommendation of a different room to study if the current study room is too hot, too humid or with deficient air quality (too crowded and/or poorly ventilated), when she is at the university and not in her lunch break. For example, we can imagine that she is initially studying in the Room *Tomás Pollán* in the *Betancourt* building, but within a few minutes the room starts getting crowded, raising the CO_2 level in the room, so she is recommended to go to an alternative study room belonging to the main library (taking into account its current air quality, temperature and humidity). For the mobile app to know specifically where Alice is located, she scans a QR code at the entrance of each room (as it was required in the university during the COVID-19 pandemic) and the app queries data about the identified room through the *sensoriZAR* (Smart Cities Lab, 2021–2023) server, which monitors the university buildings using a set of sensors deployed along the campus.

Evaluation in the indoors use case scenario

In this scenario, the data about the different rooms and buildings in the university campus are provided by the *sensoriZAR* (Smart Cities Lab,

Table 4
Recommendation triggering rules defined by Alice in the outdoors use case scenario.

Name	Condition	Recommendation type activated
LunchOutside	*AtApartment* and *Time4Lunch*	*Restaurants*
GoToShow	*AtApartment* and not *Time4Lunch* and not *Time4Dinner* and *ShowDay*	*Buildings for shows*
GoShopping	*AtChueca* and *isMadridWeekend* and not *Good4Tourism*	*Shops*
VisitMuseums	*Good4Museums* and *InTheMorning* and not *Time4Lunch*	*Museums*
VisitPlacesOfInterest	*Good4Tourism* and *AtPalacioReal*	*Points of Interest*
GoParty	*Time4Party* and *AtChueca*	*Entertainment venues*
LeisureActivities	not *Time4Lunch* and *not Time4Dinner*	Leisure

Table 5
Context rules defined by Alice in the indoors use case scenario.

Name	Type	Description
TooHotToStudy	server-based	The room's temperature is above 29 °C
TooHumidToStudy	server-based	The room's humidity is above 40%
BusyRoom	server-based	The room's CO_2 level is above 800 ppm
Time4Lunch	time-based	Alice's lunch time is between 13:00 and 15:00
AtUniversity	location-based	Alice is in the university campus *Río Ebro*

Table 6
Recommendation triggering rules defined by Alice in the indoors use case scenario.

Name	Condition	Recommendation type activated
ChangeRoomBcTooHot	*TooHotToStudy* and *AtUniversity* and *not Time4Lunch*	StudyRoom
ChangeRoomBcTooHumid	*TooHumid* and *AtUniversity* and *not Time4Lunch*	StudyRoom
ChangeRoomBcTooBusy	*BusyRoom* and *AtUniversity* and *not Time4Lunch*	StudyRoom

2021–2023) server at the University of Zaragoza. To simulate context changes, we defined a set of 5 user contexts (compiled in a JSON file); as in the test for the outdoors scenario, a new context was read and sent to Siddhi every 30 s. The results obtained when simulating the 5 context rules are as follows. First, in context 0, no recommendation was activated, as the conditions of Alice's current room were suitable for studying. In context 1, the recommendation of an alternative room was activated, as the temperature was too high. In context 2, no recommendation was activated, even though the temperature was high, because this happened during Alice's lunch break. In context 3, the recommendation of an alternative room was activated, as the CO_2 level exceeded 800 ppm. Finally, in context 4, no recommendation was activated because Alice was not in the university campus anymore.

In order to easily force changes in the context conditions, for evaluation purposes, this scenario was simulated. The requirement for this indoors use case to work in a real environment is that there is a server able to provide sensor data about the room where the user is located; specifically, as commented before, we have integrated our prototype with the functionality provided by the *sensoriZAR* (Smart Cities Lab, 2021–2023) server at the University of Zaragoza, so R-Rules is able to interact with that server in order to obtain the data needed.

5.2. Evaluation of the Suitability of the Types of Rules Supported

In this section, we present a user survey where we collected information about potential user needs. Our goal was to assess if our proposed system, with the current support of context rules and recommendation triggering rules, can satisfy all those user requirements collected. A total of 86 users answered the survey, comprising members of the Department of Computer Science and Systems Engineering of the University of Zaragoza (faculty members) as well as students of the Bachelor's Degree in Informatics Engineering of the School of Engineering and Architecture at the University of Zaragoza, in Spain. The survey contained a brief description of what the user was asked to provide: 3 examples of situations where they would appreciate receiving suggestions in their mobile phone. Besides, it included a few examples of potential answers. For more details about the contents of the survey, please see Fig. B.5 within Appendix B.3.

Some examples of answers received include *"If it is morning and I am in the office, I want the app to tell me about nearby places to have a quick lunch"* and *"If I am in a coastal city in summer and it is a sunny day, I want the app to recommend the best surrounding beaches"*. In the first example, it is important to have the ability to define location-based rules and time-based rules, whereas in the second case we also need weather-based rules and calendar-based rules. Based on the answers received through the user survey, we identified different types of rules required and types of items that users find relevant.

From the 86 answers received, 1 was discarded because it contained invalid data. In practice, we received a total of 253 examples (instead of 255, as one example was not valid and another one was empty). Most of the types of context rules required to satisfy the needs of the users that answered the survey are properly covered by R-Rules. Among the types of context rules supported by our prototype, the most-frequently required rules are location-based context rules (needed in 29.02% of the examples provided by users), followed by calendar-based context rules (needed in 14.51% of the examples), time-based context rules (needed in 13.33%), server-based context rules (5.10%), and weather-based context rules (4.31%). Based on the user survey, we also noted that about 5.88% of the examples could be easily expressed if *activity-based context rules* were available; thus, in these examples, the users would appreciate receiving recommendations of certain type when they are doing specific activities (e.g., when they are walking, cooking, driving a car, reading the email, or even procrastinating), for which human-activity recognition techniques could be applied (Jobanputra et al., 2019; Nweke et al., 2018). Furthermore, some examples (1.18%) required the triggering of recommendations based on measures provided by either their smartwatch (like their hear rate or irregular sleep patterns) or other device's sensors (like the battery level); these conditions could be expressed as server-based context rules, as there should be a server/API to query these data. Finally, in one example a user proposed to receive recommendations when he/she was bored (for this, *emotion-based context rules* could be useful). We also noticed that a number of examples provided by the users that answered the survey are actually better suited to a pull-based recommendation scenario, as there are no clear conditions that should activate a recommendation process automatically (e.g., *"If I want to watch a movie, please recommend*

me a good film" is based on the user's desires); besides, in some cases, rather than indicating the desired conditions for activation of a type of recommendation, some users indicated context variables that would be useful during the recommendation process itself (i.e., context data useful for CARS). As a conclusion, our current proposal supports most of the scenarios described by the users in a suitable way. More details about the user survey are provided in Appendix B.3.

5.3. Performance evaluation

We have also conducted experiments to assess the performance of the recommendation triggering phase, which involves running the Siddhi engine on a mobile device. The device used in the experiments described in this section was equipped with a Qualcomm Snapdragon 626 processor (octa-core A53, 2.2 GHz), 4 GB of RAM, and ran Android version 8.1.0. In the following, we explain the steps taken during these performance evaluation experiments.

First, we established a set of context rules using 7 context variables: *dayOfTheWeek* (which can be Monday, Tuesday, Wednesday, etc.), *season* (winter, spring, summer, or autumn), *partOfTheDay* (early morning, morning, afternoon, or night), *timeForDailyActivity* (lunch or dinner), *weatherStatus* (clear, cloudy, raining, etc.), *atHome* (which is true if the user is at home and false otherwise), and *onHolidays* (indicating if the user is on vacation).

Then, a Python script was written to generate recommendation triggering rules by randomly combining the previously defined context rules. The script avoids combinations of contradicting context rules (e.g. "weatherStatus=clear AND weatherStatus=cloudy") in a single rule, as they would never be satisfied. The script starts by including one context rule in a triggering rule and then, with a 25% probability, another context rule is added, and with a 75% probability no rule is added. This process was repeated until no more non-contradicting context rules could be added or the last iteration's random decision did not add a new rule. Using this procedure, 300 recommendation triggering rules were synthetically generated for evaluation purposes (each with a number of context rules between 1 and 7), with an average of 3.36 context rules per recommendation triggering rule.

We then used another Python script to randomly generate a set of example contexts. These contexts defined values for each of the possible context variables used in the context rules, and were formatted in JSON, as required by Siddhi. Specifically, 21 synthetic contexts were generated for experimental evaluation.

Finally, we conducted the experiments. In each experiment, we defined all context rules in the mobile device's app and a subset of the recommendation triggering rules (the first n rules, depending on the number of rules to consider in that experiment), and simulated context changes by sending each context to Siddhi (one update every 20 seconds). In each experiment, we measured the *triggering latency* for each type of recommendation triggered by a context change. The triggering latency is the time between the context change and the detection of the need to trigger a specific recommendation, and is measured using a module called *SiddhiAppManager* (for more details about this module, please see Appendix A.3). We started measuring the time just before sending the context to Siddhi. Since a context change can activate multiple recommendation triggering rules simultaneously, we recorded the triggering latency for each individual recommendation triggering rule activated by the change.

The left of Fig. 10 displays the average triggering latency, minimum triggering latency (which is the triggering latency of the first rule activated by a context change), and maximum triggering latency (triggering latency of the last rule activated by a context change). As expected, the highest triggering latency significantly increases with the number of recommendation triggering rules (within an acceptable range), but the average triggering latency only increases slightly. The minimum latency remains unaffected by the number of rules, as Siddhi evaluates them one by one, and therefore the cost of the first activation remains unchanged regardless of the number of rules. It is important to note that the latencies measured are always very small, never exceeding 220 milliseconds even in the worst case (when there are 200 recommendation triggering rules). This is considered acceptable, as it is unlikely that a mobile user will have such a high number of recommendation triggering rules and, even if that is the case, the latencies are still small. The right of Fig. 10 shows the number of recommendation triggering rules that were activated depending on the number of recommendation triggering rules defined; as expected, the more triggering rules defined, the higher the potential number of triggering rules activated due to a context change. The detailed measurements for each of the 21 contexts are shown in Table 7 (column "C" contains the context identifier, from 0 to 20), with each cell representing the triggering latency of a single activated recommendation triggering rule.

5.4. Conclusions of the evaluation

With the evaluation performed, we have verified: (1) that the approach is generic and can adapt well to a variety of use cases (illustrated through an indoors and an outdoors use case scenario) and activate the suitable types of recommendation depending on the context; (2) that the proposal supports the expected user needs (according to the user survey performed); and (3) that the performance is appropriate, showing a low latency and scalability.

We have shown that it is indeed possible to give mobile devices a more active role, by assigning them the responsibility to decide when a specific type of recommendation should be triggered, thanks to the execution of rule-based engines on the users' devices, which as far as we know has not been proposed by other researchers in the field of mobile CARS. In this way, we avoid communicating context data to the environment managers, thus safeguarding the user privacy and avoiding wireless communications.

On the negative side, in the development of this work we experience some typical difficulties that arise when building mobile apps (Ahmad et al., 2017, 2018). Specifically, keeping the prototype up-to-date was a significant challenge, due to the rapid evolution of software libraries and technologies used, with new versions and updates appearing, which in several occasions led to interoperability or compatibility issues; as mentioned in Ahmad et al. (2017), the problem of fragmentation occurs even within the same platform. Other development difficulties were related to testing, as emulators require significant computing resources and we noticed a significant difference in the experience and effort required while testing with real devices and with emulators. These difficulties are not linked to our specific approach but to the development of mobile apps in general. Creating a prototype for iOS may be a challenging development task (due to dependencies with libraries currently used), but it could be analyzed in future work.

6. Conclusions and future work

In this paper, we have presented a novel architecture for mobile devices that can trigger suitable types of recommendations when appropriate, without requiring explicit requests from the user. A crucial advantage of our proposal is that the rule-based decision engine is run on the mobile device, along with the user interface, which not only enhances the performance but also protects the user's privacy by locally evaluating the context. Additionally, our proposal empowers the user by enabling him/her to define and personalize his/her own context rules and triggering rules through a powerful user interface. We have examined the potential technologies that could be used in our prototype, and showed the feasibility of our proposal through the development of a prototype for Android devices (R-Rules), its illustration in specific use case scenarios, and an analysis of its performance and scalability. Besides, a user survey helped us to verify that the most-wanted functionalities have been suitably covered.

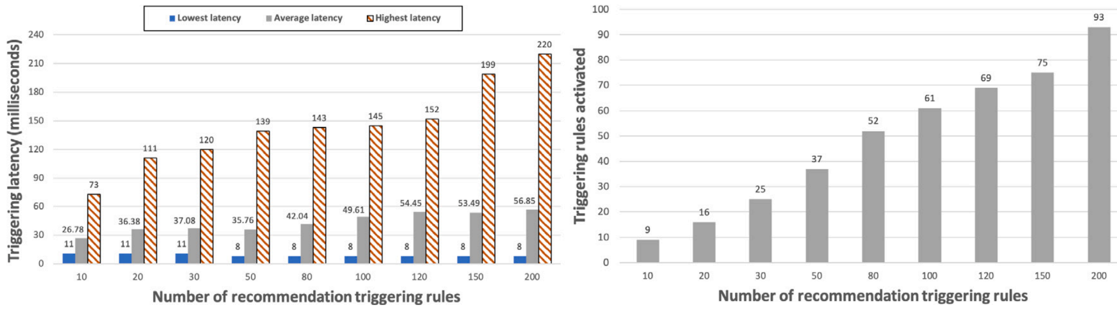

Fig. 10. Evaluation of the triggering latency (left) and number of rules activated (right).

Table 7
Individual recommendation triggering latencies measured (in milliseconds).

Number of recommendation triggering rules (from 10 to 200); C = context identifier									
C	10	20	30	50	80	100	120	150	200
0		30 66	22 21 53	9 21 29 51	9 21 29 49	9 23 30 55	9 23 30 54	9 23 31 33 38 54	14 32 43 45 52 70
1		31	25	10 22	9 22 52	9 22 40 59	16 32 50 70	16 33 48 54 75	9 23 37 43 62
2		15	14	10	13 34	16 39	8 28 45	16 36 51	14 36 52 67
3		73	64	8 51	8 34 54	17 40 50 74	21 57 69 100	11 36 46 70	15 38 47 55 71
4				13 48	14 44 51 68	9 40 47 78	13 34 49 61 83	17 34 44 51 69	8 30 41 47 49 65
5	22	17	11	19	8	11 39	9 41	8 30	12 37 42 60 61
6	25	18	15	20	9	8	17	9	8 31 47
7	13 73	19 88	14 43 61 81	10 38 56 77	9 30 46 63	18 41 62 82	19 46 68 88	9 31 49 69	15 41 62 81
8		16	20	12	15 60	10 56	11 53 61 84	9 45 52 71	15 57 64 85
9	11	11	14 80	11 41 80	10 34 58 81	14 36 59 80	20 45 68 95	18 43 65 87	9 31 52 72
10			48	20 44	11 41	13 39	12 39	11 43	10 28 41
11	13	11	13	20	14	11 78	19 88 89 90	20 87 88 89	10 54 74 75 75
12	13	12	11	12 43	10 36 63	9 31 54 76	10 31 55 83	9 32 52 73	9 31 51 72
13			16	17	17 54 67	21 58 73	22 57 71	11 40 55	13 43 59 67 90
14			22 60	16 61	27 83 95	20 65 76	16 45 65 76	18 44 58 68	18 41 59 70
15	14	14	14	23	17	23 76	13 63	20 68 106	20 45 63 99
16			24	30	20	26 120	19 107	20 103	13 48 75 89
17				36	25	35 89	34 86	28 87	18 62 92 116
18				32 80	32 72 84 94	34 83 97 108	32 82 95 105	28 87 101 113	36 101 114 126
19		111	120	49 139	36 76 87 143	36 81 92 145	34 87 99 114 152	33 86 98 115 117 155	35 96 110 128 130 168
20	57	50	61	65	48	63 121	62 128	45 115 199	46 127 173 220

As a future direction, we plan to consider the design and implementation of specific types of recommendation approaches for different scenarios, based on the concept of Side-CARS –SocIal-Distance prEserving CARS– (Ilarri et al., 2020), which take into account context variables such as the social distancing between users or the potential crowdedness/congestion of specific areas when providing recommendations. This would require integrating and evaluating specific recommendation algorithms appropriate for a variety of situations.

CRediT authorship contribution statement

Sergio Ilarri: Conceptualization, Methodology, Investigation, Writing – original draft, Supervision, Funding acquisition, Project administration. **Raquel Trillo-Lado:** Conceptualization, Investigation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Suitable references to the data used are provided in the paper. Additional info and sharable artifacts will be made available in the R-Rules web page (http://webdiis.unizar.es/~silarri/prot/RRules/).

Acknowledgments

We acknowledge the financial support of our research projects, particularly the NEAT-AMBIENCE project (PID2020-113037RB-I00, funded by MCIN/AEI/ 10.13039/50110001103). We sincerely thank Irene Fumanal (former student at the University of Zaragoza) for her intensive work on the R-Rules prototype and for her support. We are also grateful to our co-authors in related previous papers, our colleagues in the NEAT-AMBIENCE project (http://webdiis.unizar.es/~silarri/NEAT-AMBIENCE) for useful discussions, as well as our colleagues in the Smart Cities lab (https://smartcities.unizar.es) for their work on sensoriZAR. Furthermore, we thank all the people that participated in the user survey. We also thank Riadh Karchoud for clarifying some questions concerning details about the LLA prototype. Last but not least, we thank the reviewers for their useful suggestions.

Funding

This publication belongs to the project PID2020-113037RB-I00, funded by MCIN/AEI/ 10.13039/501100011033. We also acknowledge the Government of Aragon (COSMOS research group; last group reference: T64_23R), as well as our previous 2021 Aquitaine-Aragon project PASEO 2.0 (AQ-8), funded by the Government of Aragon, where an initial part of this work started.

Appendix A. Additional information about the R-rules prototype

In this appendix, we provide more information about R-Rules. Firstly, in Appendix A.1, we summarize existing approaches for the development of mobile apps, emphasizing the one chosen for this work. Secondly, in Appendix A.2, we explain the different technological solutions that we have considered as potential alternatives for rule-based context detection, justifying our choice. Finally, in Appendix A.3, we present the key packages of the software architecture of the mobile app prototype and relevant aspects about the communication with Siddhi.

A.1. Mobile development platforms

In this appendix, we provide a very brief overview of approaches that can be followed for the development of mobile apps. There are three main alternatives for the development of applications for mobile devices: developing a native app, targeting a mobile web app, or building a hybrid mobile app using a mobile development framework. Among them, we currently use React Native (Eisenman, 2015) in our R-Rules prototype, as explained in Section 4.2.

The most direct approach is to develop a *native mobile app* for a specific platform, such as Android or iOS. The main advantage is that it allows direct access to the resources and functionalities of the target platform, making the app more optimized and efficient. However, the disadvantage is that separate implementations need to be developed and maintained for each platform, which can be time-consuming and require additional human resources.

Another option is to develop a *mobile web app*, that can be run on any mobile platform with a suitable web browser, without the need for platform-specific development and maintenance. The app can be built using standard web technologies such as HTML5, CSS, and JavaScript, which makes it easier for developers to create and deploy the solution. However, the main disadvantage is that the performance and functionality may be limited compared to a native app, as it relies on the capabilities of the web browser on the user's device.

Finally, it is possible to develop a *hybrid mobile app*, based on the use of a mobile development framework. Popular frameworks include Apache Cordova (https://cordova.apache.org), Ionic (https://ionicframework.com), React Native (Eisenman, 2015) (https://reactnative.dev), Xamarin (https://dotnet.microsoft.com/apps/xamarin, now integrated as part of .NET), Flutter (https://flutter.dev), Framework7 (https://framework7.io), and Appcelerator Titanium (https://www.appcelerator.com), to cite some popular examples. This approach aims to provide the best of both worlds, by leveraging the power and flexibility of web technologies while also enabling access to platform-specific features such as sensors and the native user interface (so we can obtain the typical look-and-feel of that specific type of operating system and device). This helps to strike a balance between development cost and versatility, allowing for a cost-effective solution that can still deliver high-quality results.

In summary, the choice of using a hybrid mobile app development framework provides a good compromise between flexibility and development cost. These frameworks facilitate the collection and utilization of data from various sensors in the mobile device (such as the GPS, accelerometer, gyroscope, and others). These sensor data are essential in creating context-aware mobile recommendation systems, as they help to determine the current context and situation of the user (Ilarri et al., 2015). Having an easy access to these data simplifies the process of collecting and utilizing them, making the development of context-aware recommendation systems easier. Among these frameworks, React Native (Eisenman, 2015) is a popular choice for cross-platform mobile development, due to its functionalities, such as easy programming with JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide) and React (https://reactjs.org), hot reloading (developers can make changes to the app and see those changes reflected in real-time without having to recompile the entire app, while keeping the current state of the app) and support for wrapping native code (it is possible to incorporate platform-specific code in the app to access functionalities that are specific to the platform it is running on). In terms of interest, this framework was found to have the highest score in an online survey questionnaire presented in Biørn-Hansen et al. (2020). Due to all these reasons, we selected React Native for our prototype.

A.2. Technologies for rule-based context detection

In this section, we provide some details about the different technological solutions that we have considered as potential alternatives for rule-based context detection. In Table 1 (within Section 4.1), we already provided a summarized comparative analysis of these technologies.

A.2.1. Esper and Asper

Esper (https://www.espertech.com/esper) is a well-known and widely used CEP technology. It is open source and compatible with Java and .NET, making it accessible to a wide range of developers. Despite being first released in 2006 (Esper 0.7.0 Alpha was released on January 16, 2006), it is still being actively developed and updated (the latest version at the time of writing, which was 8.9.0, was released on May 2, 2023). Esper's rule-definition language, named *Event Processing Language* (*EPL*), is based on SQL, which makes it easy to learn and use, and offers a rich and user-friendly syntax.

In 2013, Esper was adapted for use on Android mobile devices through the project *Asper* (https://github.com/mobile-event-processing/Asper). This adaptation addressed limitations caused by missing Java packages and classes in Android. However, since then, the Asper project has not received any updates and remains based on Esper 4.8.0, which was released on January 4, 2013. This presents a significant disadvantage, as using Esper on Android would require utilizing a significantly outdated version of Esper.

A.2.2. Siddhi

Siddhi (https://siddhi.io) is a more recent open-source CEP technology, developed in Java and licensed under Apache License. The first version (Siddhi 2.2.0) was released on April 24, 2015, and the latest version at the time of writing (Siddhi 5.1.27) was released on February 1, 2023. It is well-documented and integrated with the open-source WSO2 Enterprise Integrator –WSO2 EI– (https://wso2.com/integration), belonging to its *Streaming Integrator Tooling*. It supports defining rules using the *SiddhiQL* language, which has a syntax similar to SQL, and can be used as a Java or Python library or as a microservice in Docker or Kubernetes. Additionally, it provides a desktop tool, *WSO2 Integration Studio* (https://wso2.com/integration/integration-studio), developed by WSO2, which allows for the definition and debugging of rules and the simulation of events for testing purposes.

Siddhi can be integrated into an Android app. For example, a case study on remote patient monitoring showed how Siddhi can be used to detect complex events directly on "the edge" of the network (Dhillon et al., 2018): instead of transmitting sensor data continuously, a complex event is only sent to the remote IoT Hospital Server (IHS), using the MQTT (Message Queuing Telemetry Transport) protocol, when it is

detected by the CEP engine running on the mobile device; this allows for more efficient communication of sensor data, as only complex events detected are communicated, thus reducing the amount of data transmitted.

A.2.3. Apache Flink

Apache Flink (https://flink.apache.org) is a widely-used open-source technology for processing data streams, which includes a library for handling complex events, named *Apache Flink CEP*. Although it offers CEP support, Apache Flink is not specifically a CEP engine. The latest release at the time of writing was version 1.17.1 (published on May 19, 2023) and the first version was 0.6 (published on August 26, 2014).

Apache Flink's primary focus is on distributed data processing and it has not been developed to be executed on mobile devices. Currently, there is no information on executing it on Android and, as far as we know, no efforts to port Apache Flink CEP for Android have been made.

A.2.4. Drools

Drools (https://www.drools.org) is a well-known open-source Business Rules Management System (BRMS) that uses Java technology. Business rules can be defined using the language *DRL* (*Drools Rule Language*) in .drl text files, consisting of conditions (when) and actions (then). The latest version of Drools at the time of writing was 9.44.0 (published on September 6, 2023) and the first version published in GitHub was 5.2.0.M2 (published on April 21, 2011). However, there are even previous versions from the time when this technology was part of JBoss Tools – Eclipse plugins for JBoss technology – (https://tools.jboss.org); for example, version 3.0.6 was published on August 11, 2007. According to Warszawski (2020), Drools requires more resources compared to Esper or Siddhi.

It is not feasible to run Drools natively on an Android device. Although the authors in Schinle et al. (2017) mention using a ported version of Drools for Android in the context of smart home systems, this seems to be a custom solution implemented by the authors and not publicly accessible. Challenges of porting Drools concerning memory usage are briefly mentioned in Mukherjee (2017).

A.2.5. Microsoft StreamInsight

Microsoft StreamInsight (Ali et al., 2011; Kazemitabar et al., 2010; Microsoft, 2012) is a commercial and well-known CEP technology, which requires a SQL Server license. Instead of defining rules, queries are expressed using the LINQ (.NET Language-Integrated Query) language (Krishnan et al., 2012; Marguerie et al., 2008; Meijer et al., 2006) to process streaming data.

Microsoft StreamInsight is part of Microsoft SQL Server since SQL Server version 2008 R2 (published on July 20, 2010) and is not designed to be run on mobile devices. The latest version of SQL Server at the time of writing was SQL Server 2022 (version 16.0.4065.3, published on August 10, 2023). However, it is difficult to know when the last significant updates to Microsoft StreamInsight's functionalities were performed; indeed, the latest specific version of Microsoft StreamInsight that we have found mentioned in the web is version 2.3 (part of SQL Server 2014, that was released for the first time on June 5, 2014).

A.2.6. Conclusion: Choosing between Asper and Siddhi

As only Asper and Siddhi could be executed on a mobile device, we analyzed both options in detail to determine the most suitable technology for our prototype. To make the choice, we first created a simple Android app with three buttons: one button to start the CEP engine, another one to stop it, and the third one to send an event to the engine. The following observations can be made:

- *Implementation with Asper/Esper*. A version of the sample mobile app was created by integrating the Asper library into an Android Studio project and was functional. However, as explained in Appendix A.2.1, Asper is based on Esper 4.8, which is outdated. We made some efforts to test more recent versions of Esper (such as versions 7.1 and 8.7), to verify if they were directly compatible with Android (bypassing the use of Asper). However, we found issues with missing dependencies (required classes that were not available in Android).
- *Implementation with Siddhi*. The implementation of the sample mobile app with Siddhi was smooth and no issues were found. Both version 5.1.19 and version 4 of Siddhi were tested. The ability to use and define custom functions in JavaScript, to process context data (e.g., to compare location coordinates) and use them as filters in the rules, would have been useful; however, the JavaScript Extension of Siddhi (https://docs.wso2.com/display/SIDDHIEXTENSIONS/JavaScript+Extension), which enables this feature, is not well supported on Android, as some classes are not available. As a result, the rules and computations had to be defined with SiddhiQL, using a more complex and verbose syntax.

As a conclusion, we chose Siddhi as the CEP technology to be used in our prototype. There are some surveys that have analyzed these and other CEP technologies (Dayarathna & Perera, 2018; Giatrakos et al., 2019). However, these studies do not have a focus on the execution of these technologies on mobile devices for use in the context of mobile recommender systems.

A.3. Packages in the mobile app prototype and communication with Siddhi

The mobile app prototype is composed by several key packages, as shown in Fig. A.1, where we can see that part of the code was developed using native code (in Java) and part of the code using JavaScript. Thus, the main code of the mobile app is written in JavaScript, but we need to use Android's native code to access Siddhi.

The package containing the part developed in JavaScript is composed by several subpackages in charge of different tasks: (1) *screens*, which contains the code needed to create the graphical user interface and is divided in several subpackages to manage context rules, triggering rules and exclusion sets; (2) *realmSchemas*, which contains the database schemas and the code needed to handle them; (3) *em*, which is responsible for the tasks related to the communication with the environment managers; (4) *siddhi*, which handles the conversion between rules expressed in the graphical user interface and rules written according to the syntax of Siddhi; (5) *exclusionSets*, which checks and evaluates the exclusion sets; and (6) *event*, with operations related to the computation of the context (location, etc.).

To handle the communication between the native code and the code in JavaScript, we created an Android Native Module (https://reactnative.dev/docs/native-modules-android), called *SiddhiClienteModule*, operating as a client of the Siddhi engine: it invokes operations offered by Siddhi (e.g., start the engine, stop it, send an event, or retrieve a result) from the main code of the mobile app in JavaScript. There are three main native classes in our implementation: (1) the *SiddhiAppManager* manages the communication between the app and the Siddhi engine, (2) the *SiddhiService* implements the logic of the bound service, and (3) the *SiddhiClientModule* acts as a bridge between the Android code and React Native code, allowing operations described in this module to be called from the JavaScript code of the mobile app.

The mobile app updates the Siddhi engine with context information every specified *context update period* (e.g., every 30 seconds), unless there are no changes in the context. It also needs to listen for potential results from Siddhi, that may trigger the execution of a recommendation process of a specific type of item (e.g., the user could receive restaurant recommendations when the lunch break period starts, if the user has defined and activated that recommendation triggering

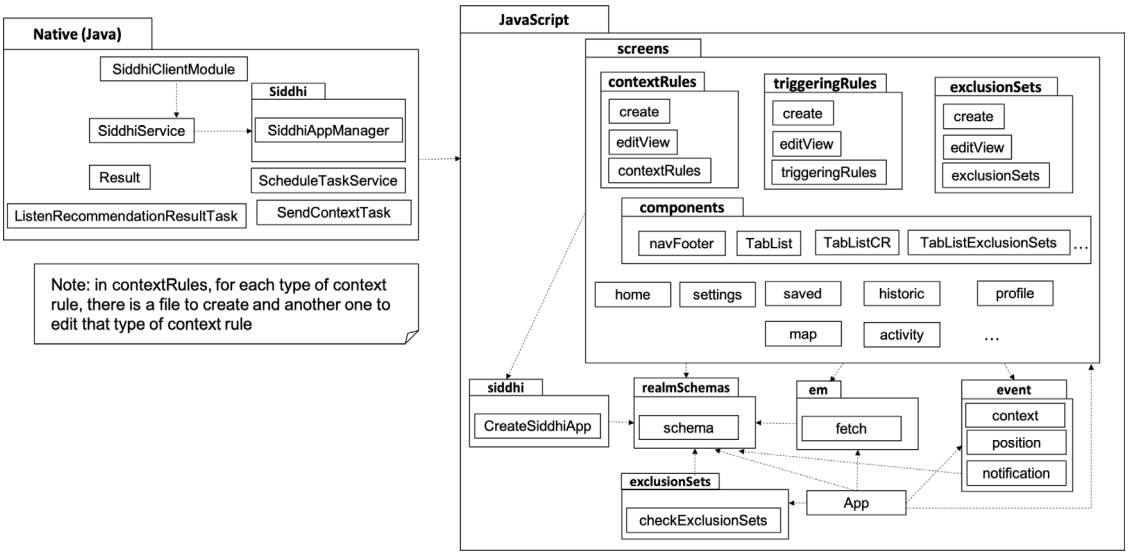

Fig. A.1. Package diagram of the mobile app prototype.


```
{"UserContext":{
   "contextId":"5",
   "date":"16/12/2022",
   "time":"15:05:00"},
"Weather":{
   "weather": [
      {
        "main": "Rain"
      }
    ],
   "main": {
      "temp": 16.12
   }
},
"Location" : {
   "coords":
   { "longitude": -3.712985,
     "latitude": 40.415312 } }
}
```


(a) *Context 5*


```
{"UserContext":{
   "contextId":"6",
   "date":"16/12/2022",
   "time":"15:45:00"},
"Weather":{
   "weather": [
      {
        "main": "Rain"
      }
    ],
   "main": {
      "temp": 16.12
   }
},
"Location" : {
   "coords":
   { "longitude": -3.712985,
     "latitude": 40.415312 } }
}
```


(b) *Context 6*


```
{"UserContext":{
   "contextId":"7",
   "date":"16/12/2022",
   "time":"17:30:00"},
"Weather":{
   "weather": [
      {
        "main": "Clouds"
      }
    ],
   "main": {
      "temp": 16.12
   }
},
"Location" : {
   "coords":
   { "longitude": -3.714994,
     "latitude": 40.416838 } }
}
```


(c) *Context 7*

Fig. B.1. Examples of contexts in the outdoors use case scenario (JSON format).

rule). For this, we use React Native's *Headless JS* (https://reactnative.dev/docs/headless-js-android), which enables the execution of tasks in JavaScript while the app is running in the background, to create two services. The first one, named *SendContextTask*, periodically sends the user's context data to Siddhi and has been defined based on (Silva, 2019). The second one, named *ListRecommendationResultTask*, listens to results provided by Siddhi; an auxiliary thread is created by the *SiddhiClientModule* to retrieve the results from Siddhi to avoid interrupting its main executing thread by a task that might require some time to complete.

Appendix B. Additional details about the experimental evaluation

In this appendix, we provide some additional details about the experimental evaluation performed. Firstly, in Appendix B.1, we show some examples of contexts used in the experimental evaluation. Secondly, in Appendix B.2, we describe the dataset of items that we used to illustrate that our approach was working well in the outdoors scenario as well as some examples of recommendations obtained during that evaluation. Finally, in Appendix B.3, we present complementary information about the user survey, including its specific content.

B.1. Examples of contexts used in the evaluation

First, we provide some examples of contexts defined for evaluation. On the one hand, Fig. B.1 shows three examples of contexts defined in the outdoors use case scenario presented in Section 5.1.1. On the other hand, Fig. B.2 illustrates three other examples of contexts for the indoors use case scenario described in Section 5.1.2. Finally, Fig. B.3 presents an example of context defined for the performance evaluation shown in Section 5.3.

B.2. Dataset and examples of recommendations for the outdoors scenario

To illustrate the use of our proposal in the outdoors scenario (described in Section 5.1.1), we needed data about items to recommend and also user ratings for those items. For this purpose, we extracted data from the Madrid's Open Data Portal (https://datos.madrid.es/portal/site/egob), which provides data in different formats, such as JSON and XML. For each item, we recorded the following attributes: (1) *id* is a unique identifier generated by us; (2) *title* is the name of the item; (3) *description* is a short text describing the

```
{"UserContext":{
   "contextId":"0",
   "date":"26/07/2022",
   "time":"10:00:00"},
"Weather":{
   "weather": [
      {
        "main": "Rain"
      }
    ],
   "main": {
      "temp": 13.12
   }
},
"Location" : {
   "coords":
   { "longitude": -0.8884599384953313,
     "latitude": 41.683260908334546 } },
"Sensorizar" : {

      "co2": [
        {
           "ts": 1658981136975,
           "value": "623"
        }
      ],
      "temperature": [
        {
           "ts": 1658981870601,
           "value": "27.08"
        }
      ],
      "humidity": [
        {
           "ts": 1658981870601,
           "value": "38.57"
        }
      ]
   }
}
```


(a) *Context 0*


```
{"UserContext":{
   "contextId":"1",
   "date":"26/07/2022",
   "time":"11:00:00"},
"Weather":{
   "weather": [
      {
        "main": "Rain"
      }
    ],
   "main": {
      "temp": 13.12
   }
},
"Location" : {
   "coords":
   { "longitude": -0.8884599384953313,
     "latitude": 41.683260908334546 } },
"Sensorizar" : {

      "co2": [
        {
           "ts": 1658981136975,
           "value": "623"
        }
      ],
      "temperature": [
        {
           "ts": 1658981870601,
           "value": "31"
        }
      ],
      "humidity": [
        {
           "ts": 1658981870601,
           "value": "38.57"
        }
      ]
   }
}
```


(b) *Context 1*


```
{"UserContext":{
   "contextId":"3",
   "date":"26/07/2022",
   "time":"16:00:00"},
"Weather":{
   "weather": [
      {
        "main": "Rain"
      }
    ],
   "main": {
      "temp": 13.12
   }
},
"Location" : {
   "coords":
   { "longitude":  -0.8884599384953313,
     "latitude": 41.683260908334546 } },
"Sensorizar" : {

      "co2": [
        {
           "ts": 1658981136975,
           "value": "800"
        }
      ],
      "temperature": [
        {
           "ts": 1658981870601,
           "value": "25"
        }
      ],
      "humidity": [
        {
           "ts": 1658981870601,
           "value": "38.57"
        }
      ]
   }
}
```


(c) *Context 3*

Fig. B.2. Examples of contexts in the indoors use case scenario (JSON format).


```
{
 "UserContext": {
   "hasId": "1",
   "date": "2022/06/26",
   "time": "22:31:38",
   "status": "onHolidays",
   "atHome": "yes"
 },
 "Preferences": [
   {
     "typeOf": "Education"
   },
   {
     "typeOf": "Shows"
   }
 ],
 "Observations": [
   {
     "observedBy": "sensor 513",
     "featureOfInterest": "user 1",
     "observedProperty": "WeatherStatus",
     "observationValue": "Clouds",
     "timeOfMeasurement": "04:27:49"
   },
   {
     "observedBy": "sensor 513",
     "featureOfInterest": "user 1",
     "observedProperty": "WeatherTemperature",
     "observationValue": "23",
     "timeOfMeasurement": "12:24:53"
   }
 ]
}
```


Fig. B.3. Example of context defined for the performance evaluation.

item; (4) *im* is an optional link to an image related to the item; (5) *longitude and latitude* are the geographic coordinates of the location of the item; (6) *begin and ending* (optional) are the starting and ending dates of the period when the item is available; (7) *category* is the label (among the different categories that we consider) that indicates the category the item belongs to; and (8) *subcategories* indicate the subcategories it belongs to. The different categories of items considered, along with the sources where the data can be found, are listed in Table B.1.

For illustration, and without loss of generality (as our focus is not on the recommendation algorithms but on the activation of different types of recommendations), we also integrated a couple of simple recommendation algorithms (provided by Apache Mahout) to complement our evaluation in the outdoors scenario: *Item-Based Collaborative Filtering – IBCF–* (Sarwar et al., 2001) (which suggests items with ratings similar to the ones the user has liked in the past) and a *random* baseline recommendation algorithm (that recommends items randomly). We generated user ratings to simulate different user profiles (a total of 12249 ratings were generated) and we run our previous scenario with each of these two recommenders. Alice had already rated different types of activities she enjoyed in Madrid and other cities in the past. We can summarize her preferences as follows. She likes the following types of *restaurants*: *vegan restaurants*, *vegetarian restaurants*, *tapas bars* and *bars*. She is interested in *places of interest* such as *buildings and monuments*, *cooking schools* and *wine and oil tastings*. Concerning *entertainment venues*, she loves *cocktail bars* and *live music*. She is also attracted to *shops* selling products related to *crafts and gifts*, *home* and *decoration*. For *accommodation*, she prefers staying in *hostels* and *aparthotels*. She

Table B.1
Datasets of items in Madrid considered in the outdoors use case scenario.

Category	Description	Source
Restaurants	Restaurants in the city	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=ce33a73970504510VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Accommodation	Lodging options	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=df42a73970504510VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Shops	Shops, commerce and markets	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=86e3a73970504510VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Entertainment venues	Places for entertainment	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=54d4a73970504510VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Points of interest	Points of tourist interest	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=3b70a73970504510VgnVCM2000001f4a900aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Museums	Museums in the city	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=118f2fdbecc63410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Show venues	Artistic performance venues: theaters, cinemas, film libraries, auditoriums, concert halls	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=842385ce457a8410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default
Leisure	Municipal cultural and leisure activities in the next 100 days	https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=6c0b6d01df986410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

also likes *museums* (excluding *planetariums* and *foundations*). Among the *buildings for shows*, she likes *auditoriums* and *concert halls*. Finally, she likes *leisure* activities related to *cinema and audiovisual activities*, *circus and magic*, and *dance*.

In Fig. B.4 we can see an example of recommendations for context 18, where the types of recommendations that could be activated are *Leisure*, *PlacesOfInterest*, and *Museums*; each line shown in Fig. B.4 contains several fields separated by semicolons: the ID of the item, its name, the type of recommendation, and the list of subcategories separated by comma (the rectangles indicate the subcategories that have been defined as part of the preferences of Alice in our outdoors use case scenario). The random recommender returns 3 items: 2 of type *PlacesOfInterest* (and more specifically belonging to the subcategory *cultural facilities*) and 1 of type *Leisure* (subcategory *Exhibitions*); these items are recommended randomly and indeed they do not match in this case the user preferences. However, with the IBCF recommender, the items obtained fit the user preferences: Alice obtains 7 items in the category *PlacesOfInterest* (subcategory *Buildings and monuments*).

B.3. Additional details about the user survey

As a complement to Section 5.2, Fig. B.5 shows the content of the user survey and Table B.2 shows the categories of items required by the

Table B.2
Types of items reported in the user survey.

Type of activity/item	Number of examples	Percentage
Hiking/running routes	14	16.47%
Tourist attractions	12	14.12%
Restaurants	12	14.12%
Shops	8	9.41%
Bars/pubs	8	9.41%
Leisure activities and events	8	9.41%
Shows	7	8.24%
Hotels/hostels	7	8.24%
Beaches/coves	7	8.24%
Cinemas	5	5.88%
Museums	5	5.88%
Cycling routes	5	5.88%
Public transport	5	5.88%
Sport activities	5	5.88%
Quiet and secluded places	3	3.53%
Pharmacies	3	3.53%
Concerts	3	3.53%

(*continued on next page*)

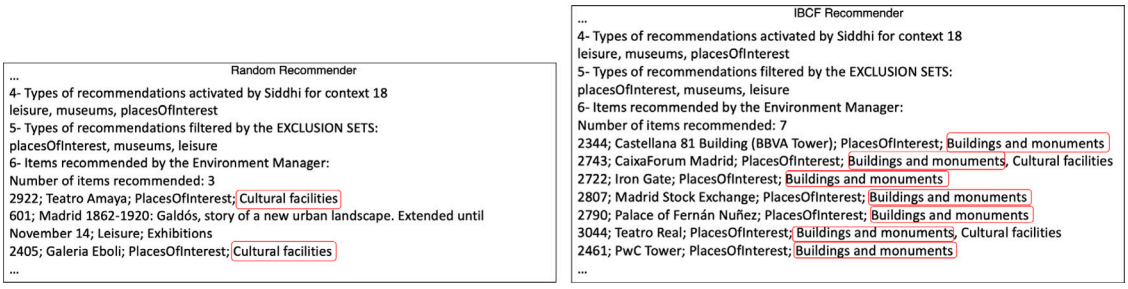

Fig. B.4. Examples of recommendations in the outdoors use case scenario: using a random recommender (left) and IBCF (right).

Recommender systems are programs that suggest things that may interest us (movies, places to visit, hotels, restaurants, books, shows, museums, etc.). Imagine that you have a mobile app available that allows you to configure under what conditions it will automatically recommend something specific to you. For example:

- *If I am out of town and it's lunchtime, I want the app to recommend a restaurant.*
- *If I am doing sports, I want the app to recommend appropriate music.*
- *If I am sightseeing and it starts to rain, I want the app to recommend something to visit protected from the rain (e.g., indoors).*
- *If it is midweek and I am home and the weather is good, I want the app to recommend outdoor sports activities.*
- *If it is Saturday and it is at night and I am not away from home, I want the app to recommend me party places.*

The above are just 5 examples. We ask you to provide 3 more, different from the previous ones, that you think would be useful to you.

Fig. B.5. Textual content of the user survey.

Table B.2 (*continued*).

Type of activity/item	Number of examples	Percentage
Bookstores	2	2.35%
Swimming pools	1	1.18%
Adventure activities	1	1.18%

users along with the relative percentage of each category in the user survey (percentage of examples relative to items of that category provided by the users). It should be noted that the types of items are related to the types of recommendations available, which is complementary to this work, as our proposal is generic enough to accommodate different types of recommendations. Anyway several popular categories among users have been considered in our use case scenarios (shops, museums, restaurants, leisure activities, etc.), which shows that those scenarios can be considered representative of typical situations.

References

Adomavicius, G., Bauman, K., Tuzhilin, A., & Unger, M. (2021). Context-aware recommender systems: From foundations to recent developments. In *Recommender systems handbook* (3rd Ed.). (pp. 211–250). New York, NY: Springer, http://dx.doi.org/10.1007/978-1-0716-2197-4_6.

Adomavicius, G., Mobasher, B., Ricci, F., & Tuzhilin, A. (2011). Context-aware recommender systems. *AI Magazine*, *32*(3), 67–80. http://dx.doi.org/10.1609/aimag.v32i3.2364.

Adomavicius, G., & Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. http://dx.doi.org/10.1109/TKDE.2005.99.

Adomavicius, G., & Tuzhilin, A. (2008). Context-aware recommender systems. In *ACM conference on recommender systems* (pp. 335–336). New York, NY, USA: ACM, http://dx.doi.org/10.1145/1454008.1454068.

Agarwal, N., Meena, C. S., Raj, B. P., Saini, L., Kumar, A., Gopalakrishnan, N., Kumar, A., Balam, N. B., Alam, T., Kapoor, N. R., & Aggarwal, V. (2021). Indoor air quality improvement in COVID-19 pandemic: Review. *Sustainable Cities and Society*, *70*, 102942:1–102942:15. http://dx.doi.org/10.1016/j.scs.2021.102942.

Ahmad, A., Feng, C., Tao, M., Yousif, A., & Ge, S. (2017). Challenges of mobile applications development: Initial results. In *Eighth IEEE international conference on software engineering and service science* (pp. 464–469). Beijing, China: IEEE, http://dx.doi.org/10.1109/icsess.2017.8342956.

Ahmad, A., Li, K., Feng, C., Asim, S. M., Yousif, A., & Ge, S. (2018). An empirical study of investigating mobile applications development challenges. *IEEE Access*, *6*, 17711–17728. http://dx.doi.org/10.1109/access.2018.2818724.

Akrivopoulos, O., Chatzigiannakis, I., Tselios, C., & Antoniou, A. (2017). On the deployment of healthcare applications over fog computing infrastructure. In *41st annual computer software and applications conference* (pp. 288–293). USA: IEEE, http://dx.doi.org/10.1109/compsac.2017.178.

Ali, M., Chandramouli, B., Goldstein, J., & Schindlauer, R. (2011). The extensibility framework in Microsoft StreamInsight. In *27th international conference on data*

engineering (pp. 1242–1253). USA: IEEE, http://dx.doi.org/10.1109/ICDE.2011.5767878.

Apple Inc (2016–2023). CarPlay. https://www.apple.com/ios/carplay. (Last Access: September 19, 2023).

Apple Inc (2018–2023). Shortcuts user guide. https://support.apple.com/en-gb/guide/shortcuts. (Last Access: 19 September 2023).

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263–277. http://dx.doi.org/10.1504/IJAHUC.2007.014070.

Beltran, V., Arabshian, K., & Schulzrinne, H. (2011). Ontology-based user-defined rules and context-aware service composition system. In *Lecture notes in computer science*: *vol. 7117*, *Extended semantic web conference (ESWC 2012) workshops* (pp. 139–155). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-25953-1_12.

Biørn-Hansen, A., Rieger, C., Grønli, T.-M., Majchrzak, T. A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, *25*(4), 2997–3040. http://dx.doi.org/10.1007/s10664-020-09827-6.

Boley, H. (2006). The RuleML family of web rule languages. In *Lecture notes in computer science (LNCS)*: *vol. 4187*, *International workshop on principles and practice of semantic web reasoning* (pp. 1–17). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/11853107_1.

Boley, H., Paschke, A., & Shafiq, O. (2010). RuleML 1.0: The overarching specification of web rules. In *Lecture notes in computer science (LNCS)*: *vol. 6403*, *International workshop on rules and rule markup languages for the semantic web* (pp. 162–178). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-16289-3_15.

Boyaci, O., Beltran, V., & Schulzrinne, H. (2011). Bridging communications and the physical world: Sense everything, control everything. In *Fifth international conference on principles, systems and applications of IP telecommunications* (pp. 1–6). New York, NY, USA: ACM, http://dx.doi.org/10.1145/2124436.2124455.

Bucchi, M., Grez, A., Quintana, A., Riveros, C., & Vansummeren, S. (2022). CORE: A complex event recognition engine. *Proceedings of the VLDB Endowment*, *15*(9), 1951–1964. http://dx.doi.org/10.14778/3538598.3538615.

del Carmen Rodríguez-Hernández, M., & Ilarri, S. (2016). Pull-based recommendations in mobile environments. *Computer Standards & Interfaces*, *44*, 185–204. http://dx.doi.org/10.1016/j.csi.2015.08.002.

del Carmen Rodríguez-Hernández, M., & Ilarri, S. (2021). AI-based mobile context-aware recommender systems from an information management perspective: Progress and directions. *Knowledge-Based Systems*, *215*, 106740:1–106740:29. http://dx.doi.org/10.1016/j.knosys.2021.106740.

Carrara, L., Orsi, G., & Tanca, L. (2013). Semantic pervasive advertising. In *Seventh international conference on web reasoning and rule systems* (pp. 216–222). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-39666-3_18.

Colombo-Mendoza, L. O., Valencia-García, R., Alor-Hernández, G., & Bellavista, P. (2017). Special issue on context-aware mobile recommender systems. *Pervasive and Mobile Computing*, *38*, 444–445. http://dx.doi.org/10.1016/j.pmcj.2017.03.002.

Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, *44*(3), http://dx.doi.org/10.1145/2187671.2187677.

Dave, K., & Varma, V. (2014). Computational advertising: Techniques for targeting relevant ads. *Foundations and Trends in Information Retrieval*, *8*(4–5), 263–418. http://dx.doi.org/10.1561/1500000045.

Dayarathna, M., & Perera, S. (2018). Recent advancements in event processing. *ACM Computing Surveys*, *51*(2), http://dx.doi.org/10.1145/3170432.

Dhillon, A. S., Majumdar, S., St-Hilaire, M., & El-Haraki, A. (2018). A mobile complex event processing system for remote patient monitoring. In *IEEE international congress on Internet of Things* (pp. 180–183). USA: IEEE, http://dx.doi.org/10.1109/iciot.2018.00034.

Eisenman, B. (2015). *Learning React Native: building native mobile apps with JavaScript*. Sebastopol, California, USA: O'Reilly Media.

Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., & Garofalakis, M. (2019). Complex event recognition in the big data era: A survey. *The VLDB Journal*, *29*(1), 313–352. http://dx.doi.org/10.1007/s00778-019-00557-w.

Hermoso, R., Ilarri, S., Trillo, R., & del Carmen Rodríguez-Hernández, M. (2015). Push-based recommendations in mobile computing using a multi-layer contextual approach. In *13th international conference on advances in mobile computing & multimedia* (pp. 149–158). New York, NY, USA: ACM, http://dx.doi.org/10.1145/2837126.2837128.

Hermoso, R., Ilarri, S., & Trillo-Lado, R. (2018). Proactive mobile CARS in action: A first step towards making sense of context rules. In *13th international workshop on semantic and social media adaptation and personalization* (pp. 69–74). USA: IEEE, http://dx.doi.org/10.1109/SMAP.2018.8501879.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A semantic web rule language combining OWL and RuleML. http://www.w3.org/Submission/SWRL. (Last Access: September 19, 2023).

IFTTT Inc (2011–2023). IFTTT (If This, Then That). https://ifttt.com. (Last Access: 19 September 2023).

Ilarri, S. (2021–2025). NEAT-AMBIENCE — Project website. http://webdiis.unizar.es/~silarri/NEAT-AMBIENCE. (Last Access: 19 September 2023).

Ilarri, S. (2023). R-Rules: Recommendation Rules! — Project website. A Rule-Based Triggering Recommendation Approach for Mobile Devices, http://webdiis.unizar.es/~silarri/prot/RRules. (Last Access: September 19, 2023).

Ilarri, S., Fumanal, I., & Trillo-Lado, R. (2021). An experience with the implementation of a rule-based triggering recommendation approach for mobile devices. In *19th international conference on advances in mobile computing & multimedia* (pp. 568–576). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3487664.3487806.

Ilarri, S., Hermoso, R., Trillo-Lado, R., & del Carmen Rodríguez-Hernández, M. (2015). A review of the role of sensors in mobile context-aware recommendation systems. *International Journal of Distributed Sensor Networks*, *2015*, 1–30. http://dx.doi.org/10.1155/2015/489264.

Ilarri, S., & Herrero, M. (2018). Towards the implementation of a push-based recommendation architecture. In *13th international workshop on semantic and social media adaptation and personalizatio* (pp. 87–92). USA: IEEE, http://dx.doi.org/10.1109/SMAP.2018.8501875.

Ilarri, S., Trillo-Lado, R., & Delot, T. (2020). Social-distance aware data management for mobile computing. In *18th international conference on advances in mobile computing & multimedia* (pp. 138–142). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3428690.3429164.

Ilarri, S., Wolfson, O., Mena, E., Illarramendi, A., & Sistla, P. (2009). A query processor for prediction-based monitoring of data streams. In *ACM international conference proceeding series*: *vol. 360*, *12th international conference on extending database technologies* (pp. 415–426). New York, NY, USA: ACM, http://dx.doi.org/10.1145/1516360.1516409.

Jobanputra, C., Bavishi, J., & Doshi, N. (2019). Human activity recognition: A survey. *Procedia Computer Science*, *155*, 698–703. http://dx.doi.org/10.1016/j.procs.2019.08.100.

Karchoud, R., Illarramendi, A., Ilarri, S., Roose, P., & Dalmau, M. (2017). Long-life application — Situation detection in a context-aware all-in-one application. *Personal and Ubiquitous Computing*, *21*(6), 1025–1037. http://dx.doi.org/10.1007/s00779-017-1077-2.

Karchoud, R., Roose, P., Dalmau, M., Illarramendi, A., & Ilarri, S. (2019). One app to rule them all: Collaborative injection of situations in an adaptable context-aware application. *Journal of Ambient Intelligence and Humanized Computing*, *10*, 4679–4692. http://dx.doi.org/10.1007/s12652-018-0846-8.

Kazemitabar, S. J., Demiryurek, U., Ali, M., Akdogan, A., & Shahabi, C. (2010). Geospatial stream query processing using microsoft SQL server StreamInsight. *Proceedings of the VLDB Endowment*, *3*(1–2), 1537–1540. http://dx.doi.org/10.14778/1920841.1921032.

Krishnan, R. R., Goldstein, J., & Raizman, A. (2012). A Hitchhiker's guide to Microsoft StreamInsight.

Liu, Q., Ma, H., Chen, E., & Xiong, H. (2013). A survey of context-aware mobile recommendations. *International Journal of Information Technology and Decision Making*, *12*(1), 139–172. http://dx.doi.org/10.1142/S0219622013500077.

Liu, G., Wang, C., Ma, X., & Yang, Y. (2021). Keep your data locally: Federated-learning-based data privacy preservation in edge computing. *IEEE Network*, *35*(2), 60–66. http://dx.doi.org/10.1109/mnet.011.2000215.

Livne, A., Tov, E. S., Solomon, A., Elyasaf, A., Shapira, B., & Rokach, L. (2022). Evolving context-aware recommender systems with users in mind. *Expert Systems with Applications*, *189*, 116042:1–116042:16. http://dx.doi.org/10.1016/j.eswa.2021.116042.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, *74*(Supplement C), 12–32. http://dx.doi.org/10.1016/j.dss.2015.03.008.

Lupton, D. (2013). The digitally engaged patient: Self-monitoring and self-care in the digital health era. *Social Theory & Health*, *11*(3), 256–270. http://dx.doi.org/10.1057/sth.2013.10.

Marguerie, F., Eichert, S., & Wooley, J. (2008). *LINQ in action*. USA: Manning Publications.

Meijer, E., Beckman, B., & Bierman, G. (2006). LINQ: Reconciling object, relations and XML in the .NET framework. In *ACM SIGMOD international conference on management of data* (p. 706). New York, NY, USA: ACM, http://dx.doi.org/10.1145/1142473.1142552.

Microsoft (2012). Microsoft StreamInsight — Building the Internet of Things. MSDN Magazine Issues, volume 27, number 3. https://docs.microsoft.com/en-us/archive/msdn-magazine/2012/march/microsoft-streaminsight-building-the-internet-of-things. (Last Access: 19 September 2023).

Mukherjee, C. (2017). *Build Android-based smart applications: using rules engines, NLP and automation frameworks*. USA: A Press.

Nweke, H. F., Teh, Y. W., Al-garadi, M. A., & Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, *105*, 233–261. http://dx.doi.org/10.1016/j.eswa.2018.03.056.

Ovadia, S. (2014). Automate the internet with "If This Then That" (IFTTT). *Behavioral & Social Sciences Librarian*, *33*(4), 208–211. http://dx.doi.org/10.1080/01639269.2014.964593.

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, *39*(11), 10059–10072. http://dx.doi.org/10.1016/j.eswa.2012.02.038.

Peng, Z., & Jimenez, J. L. (2021). Exhaled CO_2 as a COVID-19 infection risk proxy for different indoor environments and activities. *Environmental Science & Technology Letters*, *8*(5), 392–397. http://dx.doi.org/10.1021/acs.estlett.1c00183.

Pimenidis, E., Polatidis, N., & Mouratidis, H. (2018). Mobile recommender systems: Identifying the major concepts. *Journal of Information Science*, *45*(3), 387–397. http://dx.doi.org/10.1177/0165551518792213.

Portugal, I., Alencar, P., & Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, 205–227. http://dx.doi.org/10.1016/j.eswa.2017.12.020.

Rault, T., Bouabdallah, A., Challal, Y., & Marin, F. (2017). A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications. *Pervasive and Mobile Computing*, *37*, 23–44. http://dx.doi.org/10.1016/j.pmcj.2016.08.003.

Raza, S., & Ding, C. (2019). Progress in context-aware recommender systems – an overview. *Computer Science Review*, *31*, 84–97. http://dx.doi.org/10.1016/j.cosrev.2019.01.001.

Ricci, F., Rokach, L., & Shapira, B. (2022). *Recommender systems handbook* (3rd ed.). New York, NY: Springer, http://dx.doi.org/10.1007/978-1-0716-2197-4.

Sabic, A. (2016). Proactive recommendation delivery. In *10th ACM conference on recommender systems* (pp. 459–462). New York, NY, USA: ACM, http://dx.doi.org/10.1145/2959100.2959108.

Santos, A. C., Cardoso, J. M., Ferreira, D. R., Diniz, P. C., & Chaínho, P. (2010). Providing user context for mobile and social networking applications. *Pervasive and Mobile Computing*, *6*(3), 324–341. http://dx.doi.org/10.1016/j.pmcj.2010.01.001.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *10th international conference on world wide web* (pp. 285–295). New York, NY, USA: ACM, http://dx.doi.org/10.1145/371920.372071.

Schinle, M., Schneider, J., Blöcher, T., Zimmermann, J., Chiriac, S., & Stork, W. (2017). A modular approach for smart home system architectures based on Android applications. In *Fifth IEEE international conference on mobile cloud computing, services, and engineering* (pp. 153–156). Piscataway, NJ, USA: IEEE, http://dx.doi.org/10.1109/MobileCloud.2017.20.

Sharma, S., & Kaur, D. (2015). Location based context aware recommender system through user defined rules. In *International conference on computing, communication automation* (pp. 257–261). USA: IEEE, http://dx.doi.org/10.1109/CCAA.2015.7148384.

Silva, M. (2019). How to create an unstoppable service in React Native using Headless JS. https://medium.com/reactbrasil/how-to-create-an-unstoppable-service-in-react-native-using-headless-js-93656b6fd5d1. (Last Access: 19 September 2023).

Smart Cities Lab (2021–2023). *sensoriZAR — IoT platform for monitoring buildings*. Aragon Institute of Engineering Research, University of Zaragoza, https://eina.unizar.es/sensorizar. https://sensorizar.unizar.es. (Last Access: 19 September 2023).

Tanter, É., Gybels, K., Denker, M., & Bergel, A. (2006). Context-aware aspects. In *Lecture notes in computer science (LNCS): vol. 4089*, *International conference on software composition* (pp. 227–242). Berlin, Heidelberg: Springer, http://dx.doi.org/10.1007/11821946_15.

Warszawski, K. (2020). *Complex event processing for Internet of Things: open-source frameworks analysis*. University of Namur (Belgium), Available at https://researchportal.unamur.be/en/studentTheses/complex-event-processing-for-internet-of-things. (Last Access: 19 September 2023).

Welten, S., Mou, Y., Neumann, L., Jaberansary, M., Ucer, Y. Y., Kirsten, T., Decker, S., & Beyan, O. (2022). A privacy-preserving distributed analytics platform for health care data. *Methods of Information in Medicine*, *61*(S 01), e1–e11. http://dx.doi.org/10.1055/s-0041-1740564.

Wolfson, O., Chamberlain, S., Dao, S., Jiang, L., & Mendez, G. (1998). Cost and imprecision in modeling the position of moving objects. In *Fourteenth international conference on data engineering* (pp. 588–596). USA: IEEE Computer Society, http://dx.doi.org/10.1109/ICDE.1998.655822.

Wolfson, O., Sistla, A. P., Chamberlain, S., & Yesha, Y. (1999). Updating and querying databases that track mobile units. *Distributed and Parallel Databases*, *7*(3), 257–387. http://dx.doi.org/10.1023/A:1008782710752.

Xu, G., Cao, Y., Ren, Y., Li, X., & Feng, Z. (2017). Network security situation awareness based on semantic ontology and user-defined rules for Internet of Things. *IEEE Access*, *5*, 21046–21056. http://dx.doi.org/10.1109/ACCESS.2017.2734681.

Yin, X., Zhu, Y., & Hu, J. (2021). A comprehensive survey of privacy-preserving federated learning. *ACM Computing Surveys*, *54*(6), 1–36. http://dx.doi.org/10.1145/3460427.

Zhang, B., & Sundar, S. S. (2019). Proactive vs. reactive personalization: Can customization of privacy enhance user experience? *International Journal of Human-Computer Studies*, *128*, 86–99. http://dx.doi.org/10.1016/j.ijhcs.2019.03.002.