Software System for Broadcasting and Monitoring Traffic Information

Beáta Balázsi[*], Örs-Tihamér Kardos[**], Sándor Ráduly[***], Károly Simon[****]
[*] Babeş-Bolyai University, Cluj-Napoca, Romania
[**] Babeş-Bolyai University, Cluj-Napoca, Romania
[***] Babeş-Bolyai University, Cluj-Napoca, Romania
[****] Codespring LLC, Babeş-Bolyai University, Cluj-Napoca, Romania
beata.balazsi@yahoo.com
ors.kardos77@yahoo.com
sanyesz123@yahoo.com
simon.karoly@codespring.ro

***Abstract*—The purpose of the presented project is the development of a software system for broadcasting, tracking and managing traffic related information.**

The system contains a server and an Android client application for reporting and displaying traffic related events (traffic cameras, traffic jams, accidents, roadblocks, etc.). It also includes a web application for system administrators and another web application for data management and analysis.

Like other similar applications, the Sparrow project uses a predefined camera database, but it also gives opportunity for the users to report events in real-time.

The software system ensures several additional functionalities, such as visual and audio notifications about approaching events and data filtering possibilities.

The interface of the client application is intuitive and it is easy and safe to use while driving.

I. INTRODUCTION

Sparrow is a software system for broadcasting and monitoring traffic information. The project includes a client application, a web-based administration interface and a web-based data analysis and management interface. The mobile client application provides a view for reporting events and a map view for visualizing nearby events (Figure 1).

The application can be very helpful for drivers (car, bus and truck). Truck drivers already use a similar tool, the CB radio, but using these radios they cannot know the exact location of the reported events, they are not receiving any visual notifications and sometimes the system can be even annoying (e.g. while listening music, talking with passengers or during phone calls).

There are similar applications on the market, but they do not cover all the functionalities implemented by the Sparrow project. Most of these applications use only predefined databases (e.g. iSpeedCam), others are only available in specific regions or countries (e.g. Beat the Traffic). There are similar applications using social networks and giving real time event reporting possibilities (e.g. Waze), but these applications are not specialized. In general, these applications report more than traffic related events, that is why event filtering could be a real issue. Using non-specialized user interfaces while driving could be a problem, as well.

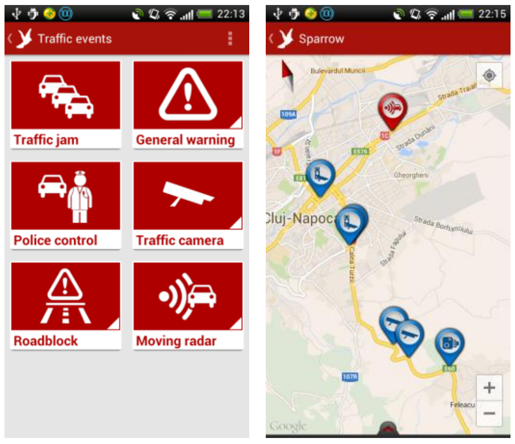


Figure 1. Event reporting and map view (Android client)

For understanding the utility of the Sparrow application let us consider an example. A car driver would like to travel through a city avoiding traffic jams. He decides to avoid the city center and he chooses to go on a bypass. Meanwhile, another driver gets into a traffic jam on the road chosen by the first driver and reports the event by using the same application. The first driver gets a notification about this new event and chooses another road.

Users can send feedbacks related to events. They can approve an event, send delete requests, attach a comment or share the event on Facebook.

Because the application is mostly used by drivers, the user interface of the mobile client provides easy navigation between different views. It has large and visible buttons with touch feedback.

The Sparrow software includes a web-based administration interface (Figure 2), which allows the system administrator to see all the available traffic events and cameras on a map. Filtering possibilities are also provided. In order to properly display large amounts of data, events are clustered. The standard clustering algorithms can be slow– for this amount of data, so the Sparrow project uses map-reduce functions to solve this problem.

The web application also provides a user management view for displaying information about users, tracks their routes and the events reported by them.

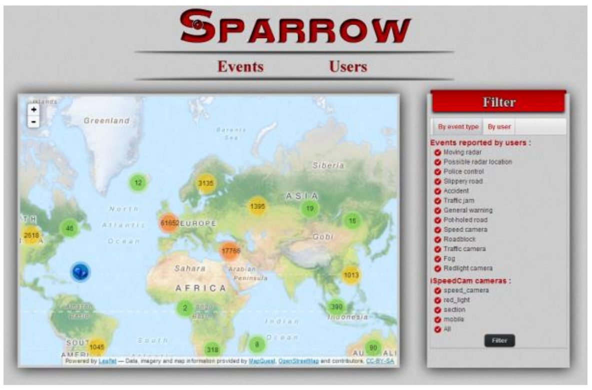


Figure 2. Map view, data clustering, filtering (Admin UI)

The data analysis interface provides optimization operations on different datasets: filtering duplicated POI-s, aggregate POI-s in a given radius, etc. The results of these operations are shown on a map and can be made persistent for a later use.

In the first section of the article the main development methods, design patterns and theoretical concepts are presented. The next sections contain a short presentation about the used technologies and development tools. These are followed by a brief description of the project, including the requirements, the architecture and the main functionalities. Finally, some conclusions and further development possibilities are presented.

II. METHODS AND DESIGN PATTERNS

The Sparrow development process is based on Agile principles, using Scrum methods and Continuous Integration practices.

The Sparrow web application is designed to manage and display large amounts of data. The project uses map-reduce functions, also supported by the used database server (MongoDB [1]). The map-reduce programming model is used to process large datasets by grouping the data by a specified key and merging all the data with the same key into one node. The method is based on two functions, the map and the reduce; if the output data structure needs to be modified, a finalize function can be also used. The map function emits key-value pairs, which are processed by the reduce function. The reduce function is used to aggregate the values with the same key. In the Sparrow project the key is composed by rounded latitude and longitude coordinates. The value is composed by the number of events grouped by category and the center of the new cluster. The map-reduce functions are called by the scheduled services provided by the Spring Framework.

The architecture of the system is designed to be easily extended with new event types. In order to achieve this, the client application gets a list from the server, containing the traffic event types. This operation is done when the user logs in. Based on this list the client application builds the event reporting view dynamically with the help of adapter classes.

The Android [2] client application is designed corresponding to the latest Android design patterns recommended by the Android operating system developers. It supports smart phones with different screen resolutions and tablets with Android, as well.

In order to provide a unified appearance on different screen sizes and pixel densities the Android client application uses 9-patch resources, which helps stretching the images without distorting them.

The system has a multi-tier architecture with a repository layer, a service layer and a presentation layer. The communication between the server and the mobile client application is realized through REST services. The server and the client use different domain models. The communication is based on the DTO (Data Transfer Object) design pattern [3]. The conversion between model objects and DTO-s is performed by assembler components.

III. TECHNOLOGIES

Sparrow is developed in Java and JavaScript programming languages, using Java technologies.

The server side is based on the Spring framework [4] [5]. The dependencies between the components are managed based on the dependency injection design pattern, by the Spring IoC (Inversion of Control) container.

The repository layer is developed using the Spring Data MongoDB framework, which provides integration with MongoDB database.

The client application for the Android platform is created using the Android SDK. The web interfaces are created with the Vaadin framework [6].

On the client side, maps are being displayed by Google Maps. The Leaflet JavaScript library is also used on the web interfaces.

For POI clustering the default Leaflet clustering algorithm is used, together with the MapReduce [7] method. The client-server communication is based on RESTful web services and it is realized by the Jersey framework, which is the reference implementation of the JAX-RS API (Java API for RESTful Services).

The SocialAuth Android library is used to connect to different social networks from Android devices. Logging is realized by the SLF4J API and the Log4j framework.

Unit tests are implemented using the JUnit framework and the Mockito framework provides mock objects for independent component tests.

IV. TOOLS

Sparrow uses Mercurial as an open source, distributed version control system. The build processes are executed with Apache Maven, an open source build and dependency management system. The central repository is managed by RhodeCode. Continuous integration is supported by Jenkins. Redmine provides a project management and issue tracking system. The XWiki, as a wiki software platform, is used to share different documents between developers.

Sparrow's code quality is maintained using SonarQube, a source code analyzer platform. It covers the main aspects of code quality: comments, coding rules, potential bugs, complexity, unit tests, duplications and architecture.

MongoDB provides the server's database. It is a document-oriented, NoSQL database management system. MongoDB supports 2d indexing, giving the possibility to write queries related to geographical coordinates. The Android client application uses SQLite database.

Apache Tomcat is used as web server. It is an open source servlet container for running Java-based web-applications.

V. THE SPARROW PROJECT

A. Requirements

The Sparrow project is separated into four main modules: the Sparrow Server, the Sparrow Admin UI, the Android mobile client and the Data Analysis UI.

The Server provides the data access layer, the business logic components, the service layer and the API for communicating with the client applications.

The Android client application provides a view for visualizing traffic events on a map and also gives the possibility for reporting events. The main functionalities provided by the Android client application:

- registration and login views;
- possibility to login using social networks (Google+, Facebook, LinkedIn);
- possibility for requesting new password (forgotten password reset);
- possibility for reporting traffic events (camera, warning, radar, traffic control, etc.);
- displays nearby traffic events on a map;
- possibility for sending feedback related to events;
- updates events on the map in real-time (inactivated events, new events, feedbacks);
- sends visual and auditive notifications about approaching events (even if the application runs in background);
- possibility for filtering events by type.

The Sparrow Admin UI gives the possibility for the administrator to manage the information about users and events from the database. The main functionalities provided by the Sparrow Admin UI application:

- login view;
- displays events and traffic cameras on a map, providing optimizations using clustering methods;
- filtering possibilities by users and event types;
- displays users' data (reported events, activity, routes etc.);

The Data Analysis module provides an interface for data source management and possibility to execute different operations for optimizing datasets. The main functionalities provided by the Data Analysis module:

- possibility to execute optimization operations on a given dataset (filter duplicates, aggregate events in a given radius, etc.);
- displays results on a map;
- possibility to save the current dataset into a database (after optimization).

B. Architecture

The Sparrow server contains the MongoDB database, the Backend and the Admin UI.

The Backend provides a Data Access Layer, realized with the Spring Data MongoDB framework. It also contains a Service Layer with business logic components. Sub-systems are communicating with the Backend via the public interfaces provided by the service components. The Model package contains the main entities. The API module is responsible for communicating with the client application via REST services.

The Admin UI communicates directly with the Backend, using service interfaces. The UI components are grouped in the View package, while the controller components are separated in the Controller package.

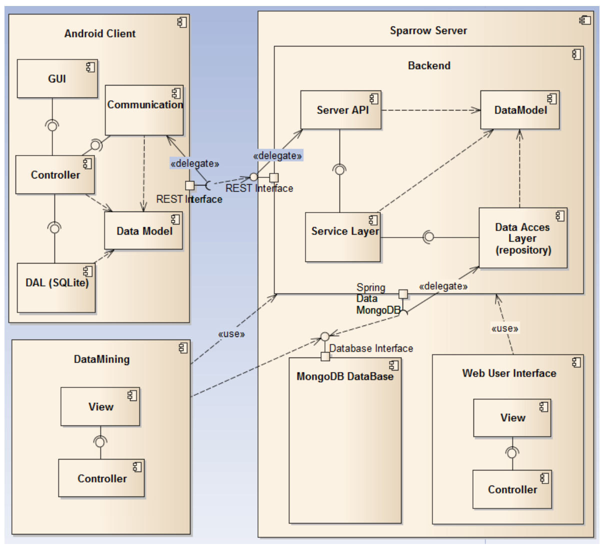


Figure 3. System architecture

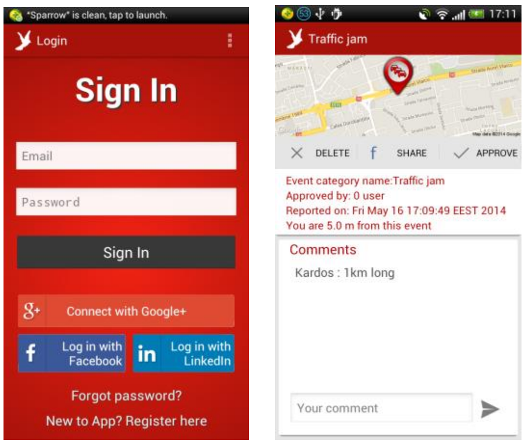
On client side SQLite database is used. The business logic is implemented by components from the Controller module. The communication with the server is based on RESTful web services. The DTO design pattern is used; DTOs are transferred in JSON format.

The DataMining module uses Backend services and communicates directly with the MongoDB database.

C. Using the Sparrow software

In order to use the Sparrow Android client application, users have to register into the system. They need to give their full name, a valid e-mail address and a password. The system sends an e-mail which contains a validation link. Once they validate their account, they can start to use the application. If someone does not want to register, he can choose to log in using social networks (Facebook, Google+, LinkedIn). If the users forget their passwords, they can claim another one by providing their e-mail address and requesting a password reset.

Once the user is logged in, he can see all the surrounding events on a map, which are represented by different markers. If they notice a traffic event they can report it on another view by choosing the corresponding category. Users can get more information about an event by tapping on the corresponding marker. They can find out the reporting date, the distance from the event and the event`s reliability number. On the same view they can approve the event, send a delete request or attach a comment to the event.



Event category name:Traffic jam
Approved by: 0 user
Reported on: Fri May 16 17:09:49 EEST 2014
You are 5.0 m from this event

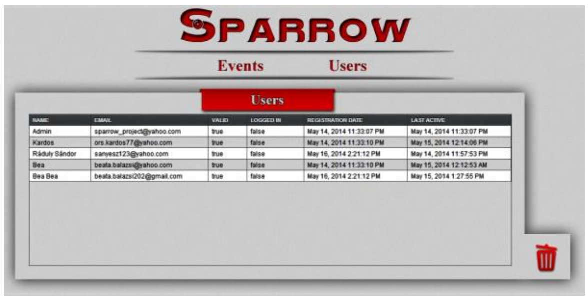
Kardos : 1km long

Figure 4. Login view and marker information view

Users can modify the information given at registration on the profile view. On the settings view they have filtering possibilities based on event categories.

After logging into the web admin application, by default the administrator can see all the events and traffic cameras on a map. In case he is interested in some specific events, or events reported by specific users, he has the possibility to filter out the other events.

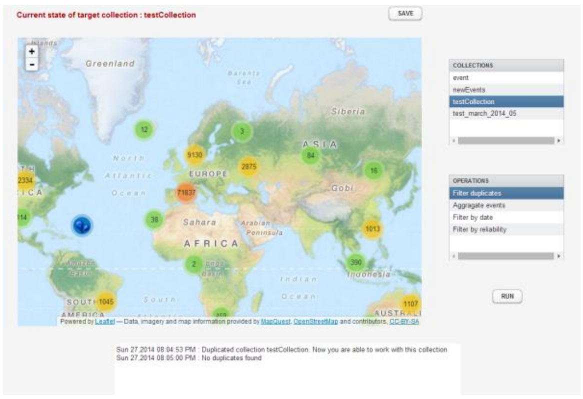
On the user management view the administrator can see all the users registered in the system and information about them.



NAME	EMAIL	VALID	LOGGED IN	REGISTRATION DATE	LAST ACTIVE
Admin	sparrow_project@yahoo.com	true	false	May 14, 2014 11:33:07 PM	May 14, 2014 11:33:07 PM
Kardos	ors.kardos77@yahoo.com	true	false	May 14, 2014 11:33:10 PM	May 15, 2014 12:14:06 PM
Ráduly Sándor	sanyesz123@yahoo.com	true	false	May 16, 2014 2:21:12 PM	May 14, 2014 11:57:53 PM
Bea	beata.balazsi@yahoo.com	true	false	May 14, 2014 11:33:10 PM	May 15, 2014 12:12:53 AM
Bea Bea	beata.balazsi202@gmail.com	true	false	May 16, 2014 2:21:12 PM	May 15, 2014 1:27:55 PM

Figure 5. User management view

The Sparrow project supports analysis and optimization methods by the DataMining UI, which gives possibility to manage different data sources.



Sun 27 2014 08:04:53 PM : Duplicated collection testCollection. Now you are able to work with this collection
Sun 27 2014 08:05:00 PM : No duplicates found

Figure 6. DataMining UI

After choosing an available data source, the administrator can run four kinds of methods: filtering by date and reliability number, filtering duplicates and aggregating events of the same type in a given radius. Results are displayed on the map and the current state of the dataset can be saved in the database.

VI. CONCLUSIONS AND FURTHER DEVELOPMENT

In its current state Sparrow is a prototype, which mainly targets the drivers' community. Users can get and share information about traffic, thereby they can plan their routes, avoiding possible obstacles and risks.

There are several further development possibilities, functionalities, which can expand the application:

- a web interface, where registered users can manage their data (e.g. reported events, routes) and they can plan their routes based on the reported traffic information;
- a web interface for media partners (e.g. radios), where they can report events (e.g. phone calls from radio listeners, reporting traffic events);
- possibility to integrate a configurable view in third party web pages (e.g. a portlet visualizing events in a specific region, integrated into a portal hosted by a media partner);
- a closer connection between the application and social networks (e.g. sharing the events on social networks);
- more effective data source management, possibility for selecting multiple data sources within the client-applications.

ACKNOWLEDGMENT

The authors would like to thank Codespring LLC for supporting the development process and ensuring the infrastructure.

REFERENCES

[1] ***, (2013) MongoDB Reference Documentation [Online]. Available: http://docs.mongodb.org/manual/ .

[2] Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura, Programming Android, 2nd ed., Sebastopol, California: O'Reilly Media, September 2012.

[3] Martin Fowler, Patterns of Enterprise Application Architecture, 1 edition ed., Addison-Wesley Professional, November 15, 2002.

[4] Rod Johnson, Juergen Hoeller and co., (2004-2012) Spring Framework Reference Documentation [Online]. Available: http://spring.io/docs.

[5] Clarence Ho, Rob Harrop, Pro Spring 3, New York: Springer Science+Business Media, Apress Media LLC, 17 April 2012.

[6] Marko Grönroos, Book of Vaadin: Vaadin 7 Edition, revision 2nd ed., Vaadin Ltd, 2014.

[7] Jeffrey Dean, Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, OSDI'04: Sixth Symposium on Operating System Design and Implementation, December 2004.