BACK-END DEVELOPMENT OF THE APPLICATION “LEARN LANGUAGE BY SONGS”

Sannikov M.A., Leontev P.N.
Scientific supervisor: Zvigintsev I.L., teaching assistant
Language supervisor: Pichugova I.L., senior teacher
Tomsk Polytechnic University, 30, Lenin Avenue, Tomsk, 634050, Russia
E-mail: max777alex@gmail.com

Mobile application development is very popular today. There are a lot of different mobile devices around the world, which work on various operating systems. Therefore, problems arise when you are trying to design an application under many platforms. The one is the application overloading causing some bad influence on performance and response time. The second problem is that there is some functionality, which is completely the same on different platforms, and it is really difficult to maintain such code. One way to solve these problems is to choose the other application architecture, for example, client-server. Such architecture has been chosen for our application Learn Language By Songs (LLBS), which is designed for two most popular platforms: Android and iOS.

The application installed on a user mobile device is called front-end. It contains the main functionality which allows interacting with the user. The server application (back-end) is responsible for the most “heavy” functions and provides simultaneous high-performance processing of the client devices requests. In the application LLBS the backend is developed using PHP-framework Symfony 2 and MySQL relational database management system. It provides the Application Programming Interface (API), which consists of the following functions:

- song lyrics search by its attributes (name, artist);
- user authorization and registration;
- getting the scores about listened songs.

Symfony PHP Web Development Framework is chosen for backend API because it is one of the most developing frameworks and it helps to think about the functionality of application instead of details of implementation. The main structure of request handling is shown in Fig. 1. Symphony aims to speed up the creation and maintenance of web applications and to replace repetitive coding tasks. It has the following structure: incoming requests are interpreted by the routing and passed to controller functions that return Response objects. Each "page" of the site is defined in a routing configuration file that maps different URLs to different PHP functions. The job of each PHP function, called a controller, is to use information from the request along with many other tools Symfony makes available – to create and return a Response object. In other words, the controller contains the code that turns HTTP request into HTTP response.

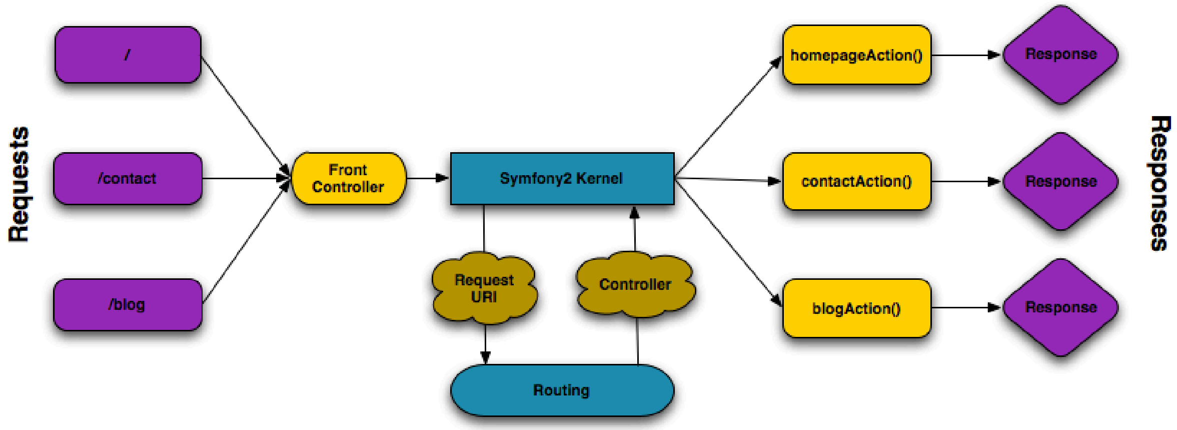

Fig. 1. Symfony 2 application structure

There are two server-side MySQL databases. The first database is responsible for storing information about user registrations. The second is a caching storage for song lyrics. The song lyric is the result of parsing the pages on popular sites with such information, for instance, azlyrics.com. Moreover, exceptional cases handling is designed for artist or song name in order to get rid of some errors. For example, if we take the string “The Beatles” and the application tries to get the song lyrics, it will fail because many sites use the pattern “artist/song_name” where the first one often does not contain the prefix “the”.

The server-side caching is necessary for reducing the load on the network and the sites with the required information about the song. The handling of incoming request begins with cache scanning. If the cache does not contain any corresponding information, then the song lyrics search will run. However, the server performance will decrease when the cache size is about a hundred of gigabytes. But the effective maximum table size for MySQL databases is usually determined by operating system constraints on file sizes, not by MySQL internal limits [1]. To solve this problem we are storing request counters for every song in the second MySQL database. It is always used in the server procedure of reducing the data which are not relevant at the given moment (shrinking). Shrinking can be performed both manually and automatically (for instance, if the cache size is more than predefined constant). In addition, front-end app also uses caching. The main reason is to decrease response time for all actions performed by the user. Client has SQLite database which stores only song's lyrics which are on user device. When user launches the application, background scanning of the file system will be run to update the list of available songs. If one of the songs is not found, all the information about it will be deleted from the cache. The full scheme of such interaction is shown in Fig. 2.

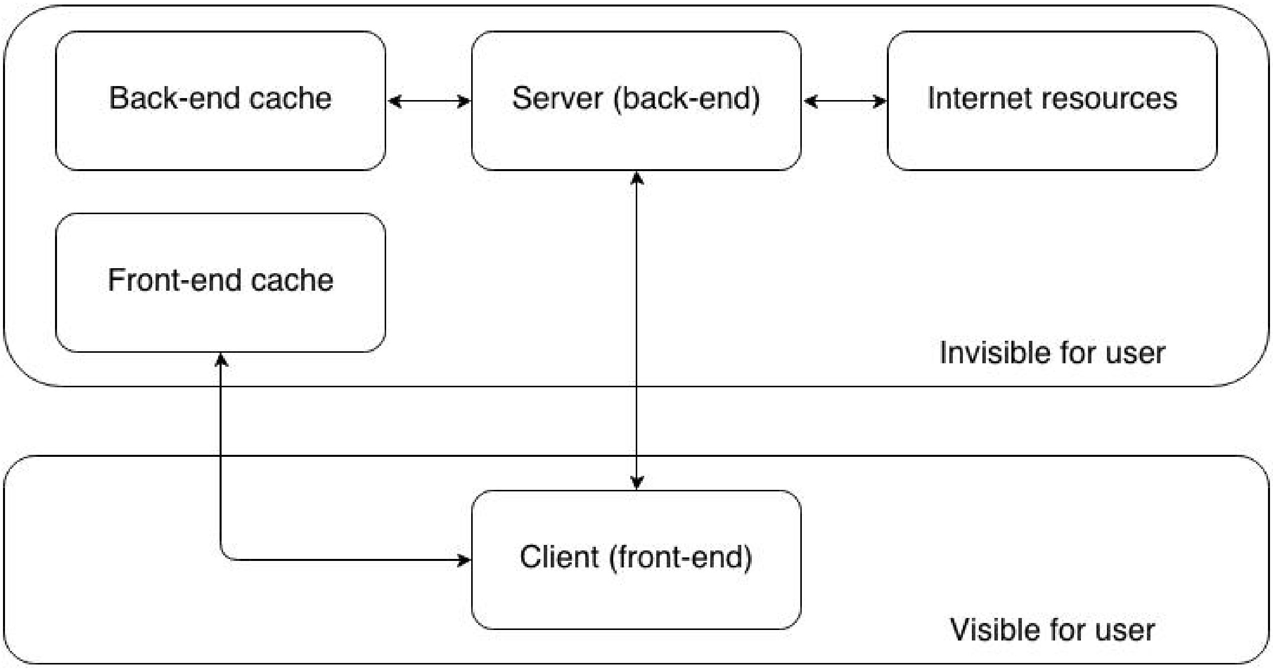

Fig. 2. Application sheme

As a conclusion we can say that back-end development is more complex task than front-end because of many server implementation features and advanced tasks such as caching, HTTP-requests processing or server protection from the overloading.

References

1. MySQL Documentation [Electronic resource] Access mode: http://dev.mysql.com/doc/refman/5.0/en/table-size-limit.html
2. Android for developers [Electronic resource], Access mode: https://developer.android.com - free.
3. Client-server model [Electronic resource]. Access mode: http://en.wikipedia.org/wiki/Client%E2%80%93server_model – free.
4. High Performance PHP framework for web-development – Symphony [Electronic resource]. Access mode: http://symfony.com/ - free
5. Application Programming Interface [Electronic resource]. Access mode: http://en.wikipedia.org/wiki/Application_programming_interface - free.
6. Mobile Development overview [Electronic resource]. Access mode: http://www.clarity-ventures.com/articles/article/676/mobile-development-overview-statistics-on-mobile-development-mobile-development - free.